III IZDANJE

Gilberto Najera-Gutierrez,
Juned Ahmed Ansari

Kali Linux

Testiranje neprobojnosti veba

Istražite motode i alatke etičkog hakovanja pomoću Kali Linuxa

kompjuter biblioteka

Packt>

Gilberto Najera-Gutierrez
Juned Ahmed Ansari

Kali Linux

Testiranje neprobojnosti veba

kompjuter biblioteka

Packt>

Izdavač:

kompjuter biblioteka

Obalskih radnika 4a, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autori: Gilberto Najera-Gutierrez
Juned Ahmed Ansari

Prevod: Slavica Prudkov

Lektura: Miloš Jevtović

Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Svetlost“, Čačak

Tiraž: 500

Godina izdanja: 2018.

Broj knjige: 502

Izdanje: Prvo

ISBN: 978-86-7310-525-3

Web Penetration Testing with Kali Linux Third Edition

Gilberto Najera-Gutierrez

Juned Ahmed Ansari

ISBN 978-1-78862-337-7
Copyright © 2018 Packt Publishing

All right reserved. No part of this book may be reproduced or transmitted in any form or by means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.
Autorizovani prevod sa engleskog jezika edicije u izdanju „Packt Publishing”, Copyright © 2018.

Sva prava zadržana. Nije dozvoljeno da nijedan deo ove knjige bude reprodukovan ili snimljen na bilo koji način ili bilo kojim sredstvom, elektronskim ili mehaničkim, uključujući fotokopiranje, snimanje ili drugi sistem presnimavanja informacija, bez dozvole izdavača.

Zaštitni znaci
Kompjuter Biblioteka i „Packt Publishing” su pokušali da u ovoj knjizi razgraniče sve zaštitne oznake od opisnih termina, prateći stil isticanja oznaka velikim slovima.
Autor i izdavač su učinili velike napore u pripremi ove knjige, čiji je sadržaj zasnovan na poslednjem (dostupnom) izdanju softvera. Delovi rukopisa su možda zasnovani na predizdanju softvera dobijenog od strane proizvođača. Autor i izdavač ne daju nikakve garancije u pogledu kompletnosti ili tačnosti navoda iz ove knjige, niti prihvataju ikakvu odgovornost za performanse ili gubitke, odnosno oštećenja nastala kao direktna ili indirektna posledica korišćenja informacija iz ove knjige.

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд,
се добија на захтев

O AUTORU

Gilberto Najera-Gutierrez je iskusni ispitivač neprobojnosti koji trenutno radi za jednog od najboljih provajdera servisa testiranja bezbednosti u Australiji. Stekao je sve važne sertifikate za testiranje bezbednosti i neprobojnosti, odnosno Offensive Security Certified Professional (OSCP), EC-Council Certified Security Administrator (ECSA) i GIAC Exploit Researcher and Advanced Penetration Tester (GXPN); takođe je magistrirao u oblasti računarstva, uz specijalizaciju u oblasti veštačke inteligencije.

Gilberto radi kao ispitivač neprobojnosti od 2013. godine i skoro dve decenije se bavi ispitivanjem bezbednosti. Uspešno je sproveo testiranja neprobojnosti na mrežama i aplikacijama nekih najvećih korporacija, vladinih agencija i finansijskih institucija u Meksiku i u Australiji.

Juned Ahmed Ansari (@junedlive) je istraživač sajber bezbednosti. On trenutno vodi testiranje neprobojnosti i tim za bezbednost u MNC-u. Radio je kao savetnik za velika preduzeća privatnog sektora i vodio njihov program sajber bezbednosti. Takođe je radio sa početnicima i pomagao im da obezbede svoj finalni proizvod.

Juned je vodio nekoliko treninga za testiranje neprobojnosti koji su bili fokusirani na učenje tehnika prikrivanja i izbegavanja grešaka u visoko bezbednim okruženjima. On se primarno fokusira na testiranje neprobojnosti, inteligenciju pretnje i istraživanje bezbednosti aplikacija. Stekao je sve vodeće sertifikate za bezbednost, odnosno GXPN, CISSP, CCSK i CISA. Uživa da sarađuje u javnim grupama i povremeno na blogovima na adresi http://securebits.in.

O RECENZENTU

Daniel W. Dieterle je autor čije su knjige o bezbednosti publikovane internacionalno, istraživač i tehnički urednik. Ima više od 20 godina iskustva u oblasti IT-a, a obezbedio je različite nivoe podrške i usluga stotinama kompanija, od malih preduzeća do velikih korporacija. Daniel je autor CYBER ARMS - Computer Security bloga (https://cyberarms.wordpress.com/) i Internet of Things bloga, koji je zasnovan na projektima i bezbednosti (https://dantheiotman.com/).

Kratak sadržaj

Sadržaj

POGLAVLJE 3

POGLAVLJE 4

UVOD

Veb aplikacije, kao i veb servisi, postali su deo svakodnevnog života – koriste se u vladinim procedurama, društvenim medijima i aplikacijama za bankarstvo, a pronaći ćete ih i u mobilnim aplikacijama koje šalju i primaju informacije korišćenjem veb servisa. Kompanije i ljudi generalno koriste veb aplikacije svakodnevno. Sama ova činjenica čini veb aplikacije privlačnom metom za krađu informacija i druge kriminalne radnje. Stoga, zaštita ovih aplikacija i njihove infrastrukture od napada je veoma važna za programere i vlasnike.

U poslednje vreme učestale su vesti iz raznih krajeva sveta o masovnom kompromitovanju podataka i zloupotrebama funkcionalnosti aplikacija za generisanje dezinformacija ili sakupljanje korisničkih informacija, koje su zatim prodavane marketinškim kompanijama. Ljudi počinju da brinu kako kompanije koriste i štite njihove informacije. Dakle, kompanije treba da preduzmu aktivne mere za sprečavanje takvog „curenja“ ili napada. To se radi na mnogim frontovima, od strože kontrole kvaliteta u toku razvojnog procesa, do PR-a i obaveštavanja medija kada je detektovan incident.

Pošto su razvojni ciklusi kraći i dinamičniji zbog upotrebe aktuelnih metodologija, povećanje složenosti u mnoštvu tehnologija je potrebno za kreiranje moderne veb aplikacije. Osim toga, zbog nasleđene loše prakse, pojedini programeri ne mogu u potpunosti da testiraju svoje veb aplikacije sa aspekta bezbednosti, s obzirom da im je prioritet da svoje proizvode isporuče na vreme. Ova složenost u veb aplikacijama i u samom razvojnom procesu stvara potrebu za profesionalcima koji su specijalizovani za testiranje bezbednosti i koji se uključuju u razvojni proces i preuzimaju odgovornost testiranja aplikacije u pogledu bezbednosti sa tačke gledišta napadača. Ovi profesionalci su penetracioni testeri.

U ovoj knjizi govorićemo o osnovnim konceptima veb aplikacija i penetracionog testiranja, opisujući sve faze u metodologiji, od dobijanja informacija za identifikovanje mogućih slabih tačaka, do eksploatisanja ranjivosti. Ključni zadatak penetracionog testera je da, kada pronađe i verifikuje ranjivost, posavetuje programere kako da isprave uočenu grešku i spreče da se ona ponovo javi. Prema tome, sva poglavlja u ovoj knjizi koja su posvećena identifikaciji i eksploatisanju ranjivosti uključuju i odeljak u kojem je ukratko opisano kako se sprečavaju i izbegavaju takvi napadi.

KOME JE NAMENJENA OVA KNJIGA

Napisali smo ovu knjigu imajući na umu nekoliko vrsta čitalaca. Studenti računarstva, programeri i administratori sistema koji žele da obogate svoje znanje o bezbednosti informacija ili oni koji žele da imaju karijeru u ovoj oblasti upoznaće neke osnovne koncepte i instrukcije koje su jednostavne, što će im omogućiti da izvrše prvo penetraciono testiranje u sopstvenim laboratorijama, a pronaći će i osnove i alatke za nastavak rada i učenja.

Programeri aplikacija i administratori sistema će takođe naučiti kako se napadači ponašaju u stvarnom svetu, koji aspekti treba da budu razmotreni za izgradnju bezbednijih aplikacija i sistema i kako se detektuje zlonamerno ponašanje. Iskusni stručnjaci na polju bezbednosti će u ovoj knjizi pronaći neke srednje teške i napredne tehnike eksploatacije i ideje kako da kombinuju dve ili više ranjivosti da bi izvršili sofisticiranije napade.

ŠTA OBUHVATA OVA KNJIGA?

Poglavlje 1, „Uvod u penetraciono testiranje i veb aplikacije“, obuhvata osnovne koncepte penetracionog testiranja, Kali Linuxa i veb aplikacija. Poglavlje započinje samom definicijom penetracionog testiranja i drugih ključnih koncepata, a zatim slede razmatranja pre uključivanja profesionalnog penetracionog testa, kao što je definisanje oblasti važenja i pravila primene. Zatim ćete pregledati Kali Linux i videti kako funkcionišu veb aplikacije, fokusirajući se na aspekte koji se više odnose na penetracionog testera.

U Poglavlju 2, „Podešavanje laboratorije pomoću Kali Linuxa“, naći ćete tehnički pregled okruženja testiranja koje će biti upotrebljeno u ostalim poglavljima. Započećemo poglavlje objašnjenjem šta je Kali Linux i opisom alatki koje on uključuje za namenu testiranja bezbedosti veb aplikacija, a zatim ćemo pogledati ranjive veb aplikacije koje će biti upotrebljene u narednim poglavljima za demonstraciju ranjivosti i napada.

U Poglavlju 3, „Izviđanje i profilisanje veb servera“, prikazane su tehnike i alatke koje koriste penetracioni testeri i napadači za preuzimanje informacija o tehnologijama koje su upotrebljene za razvoj, hostovanje i podršku ciljne aplikacije i identifikaciju prve slabe tačke koja može da bude dalje eksploatisana, jer je, ako se prati standardna metodologija za penetraciono testiranje, prvi korak sakupljanje informacija o ciljevima.

Poglavlje 4, „Nedostaci provere identiteta i upravljanja sesijom“, kao što i naslov govori, namenjeno je detekciji, eksploataciji i smanjenju ranjivosti koja se odnosi na identifikaciju korisnika i razdvajanje dužnosti unutar aplikacije. Počećemo objašnjenjem različitih mehanizama za proveru identiteta, a zatim ćemo objasniti kako ovi mehanizmi mogu da imaju nedostatke u dizajnu ili implementaciji i kako zlonamerni akteri ili penetracioni testeri mogu da iskoriste ove nedostatke.

U Poglavlju 5, „Detektovanje i eksploatacija nedostataka zasnovanih na injektiranju“, opisani su detekcija, eksploatacija i smanjenje najčešćih nedostataka injektiranja, jer je jedna od najvećih briga programera u pogledu bezbednosti da aplikacije ne budu ranjive

u bilo kojoj vrsti napada injektiranjem, bez obzira da li je to SQL injektiranje, injektiranje komande ili bilo koji drugi napad, jer to može da predstavlja veliki rizik za veb aplikaciju.

U Poglavlju 6, „Pronalaženje i eksploatisanje ranjivosti, prenosa i izvršenja skripta kroz sajt (XSS)“, opisano je šta je ranjivost prenosa i izvršenja skripta kroz sajt, šta predstavlja bezbednosni rizik, kako se identifikuje kada je veb aplikacija ranjiva i kako napadač može da iskoristi ovu ranjivost da preuzme osetljive informacije od korisnika ili da ih natera da nesvesno izvrše neke akcije.

U Poglavlju 7, „Cross-Site Request Forgery, identifikacija i eksploatacija“, saznaćete šta je Cross-Site Request Forgery napad i kako funkcioniše. Zatim ćemo opisati ključni faktor za detektovanje nedostataka koji ga omogućavaju i tehnike za eksploataciju. Poglavlje ćemo završiti savetima o prevenciji i izbegavanju ovih napada.

Poglavlje 8, „Napadi na nedostatke u kriptografskim implementacijama“, započinje uvodom u koncepte kriptografije koji su korisni iz perspektive penetracionog testera, kao što je način na koji SSL/TLS funkcioniše. Predstavićemo koncepte i algoritme enkripcije, kodiranje i heširanje, a zatim ćemo opisati alatke koje se koriste za identifikaciju slabih SSL/TLS implementacija, zajedno sa eksploatacijom dobro poznatih oblika ranjivosti. Zatim ćemo opisati detekciju i eksploataciju ranjivosti u uobičajenim kriptografskim algoritmima i implementacijama. Poglavlje ćemo završiti savetom kako da sprečite ranjivost kada koristite šifriranu komunikaciju ili kada skladištite osetljive informacije.

U Poglavlju 9, „AJAX, HTML5 i napadi na strani klijenta“, opisana je strana klijenta penetracionog testiranja veb aplikacije, počev od procesa analize sadržaja AJAX aplikacije i opisa programerskih alatki koje su uključene u modernim veb pretraživačima. Takođe ćemo predstaviti inovacije koje su unete u HTML5 i nove izazove za napadače i penetracione testere. Zatim, sledi odeljak u kojem je opisana upotreba programerskih alatki za zaobilaženje bezbednosnih kontrola koje su implementirane na strani klijenta, a poglavlje se završava savetima za prevenciju i izbegavanje ranjivosti AJAX-a, HTML-a 5 i strane klijenta.

U Poglavlju 10, „Ostali uobičajeni nedostaci u veb aplikacijama“, biće reči o direktnom pristupu neadekvatno zaštićenim objektima, uključivanju fajlova, „zagađenju“ HTTP parametra i ranjivosti otkrivanja informacija i njihovoj eksploataciji. Poglavlje ćemo završiti savetom kako da sprečite i otklonite ove nedostatke.

U Poglavlju 11, „Upotreba automatizovanih skenera u veb aplikacijama“, opisani su faktori koje treba uzeti u obzir kada se koriste automatizovani skeneri i fuzeri u veb aplikacijama. Takođe ćete saznati kako ovi skeneri funkcionišu i šta je fuzing. Zatim ćemo predstaviti primere upotrebe alatki za skeniranje i fuzing, koje su uključene u Kali Linux. Završićemo poglavlje predstavljanjem akcija koje penetracioni tester treba da izvrši nakon automatskog skeniranja veb aplikacije da bi programeru aplikacije isporučio rezultate ranjivosti.

ŠTA VAM JE POTREBNO ZA OVU KNJIGU?

Da biste dobili maksimum iz ove knjige

Da biste uspešno iskoristili ovu knjigu, treba da imate osnovno znanje o sledećim temama:

- instalacija Linux OS-a
- upotreba Unix/Linux komandne linije
- HTML jezik
- programiranje PHP veb aplikacije
- Python programiranje

Jedini hardver koji vam je potreban je računar sa operativnim sistemom koji može da pokrene VirtualBox ili drugi virtuelizacioni softver. Za specifikaciju preporučujemo sledeću konfiguraciju:

- Intel i5, i7 ili sličan CPU
- hard drajv od 500 GB
- 8 GB RAM-a
- internet konekcija

PREUZIMANJE FAJLOVA PRIMERA KODA

Fajlove sa primerima koda možete da preuzmete za ovu knjigu sa našeg sajta:
http://bit.ly/2HaoZyp

PREUZIMANJE KOLORNIH SLIKA

Takođe smo obezbedili PDF fajl koji ima kolorne snimke ekrana/dijagrama koji su upotrebljeni u ovoj knjizi. Možete da ga preuzmete na adresi:

http://bit.ly/2vuBVub

UPOTREBLJENE KONVENCIJE

Postoji veliki broj konvencija teksta koje su upotrebljene u ovoj knjizi.

CodeInText: ukazuje na reči koda u tekstu, nazive tabele baze podataka, nazive direktorijuma, nazive fajlova, ekstenzije fajlova, nazive putanje, skraćene URL-ove, korisnički unos i Twitter postove. Evo i primera: „Mnoge organizacije će osluškivati port koji nije deo fajla nmap-services.“

Blok koda je prikazan na sledeći način:

```
<?php if(!empty($_GET[,k‘])) {
    $file = fopen(,keys.txt‘, ,a‘);
    fwrite($file, $_GET[,k‘]);
    fclose($file);
  }
?>
```

Kada želimo da privučemo pažnju na određeni deo bloka koda, relevantne linije ili stavke će biti ispisane zadebljanim slovima:

```
<?php if(!empty($_GET[,k‘])) {
    $file = fopen(,keys.txt‘, ,a‘);
    fwrite($file, $_GET[,k‘]);
    fclose($file);
  }
?>
```

Svaki unos komandne linije ili ispis će biti prikazan na sledeći način:

```
python –m SimpleHttpServer 8000
```

Zadebljana slova: Ukazuju na novi termin, važnu reč ili reči koje vidite na ekranu. Na primer, reči u menijima ili okvirima za dijalog prikazane su u tekstu na sledeći način: „Ako otvorite karticu **Logs** unutar **Current Browsera**, videćete da veza registruje sve što korisnik radi u pretraživaču, od klikova i otkucaja, do promena prozora ili kartica.“

Upozorenja ili važne napomene će biti prikazivani u ovakvom okviru.

Dobra praksa
Preporuke kako da programirate kao stručnjak prikazne su ovako.

POVRATNE INFORMACIJE

Povratne informacije od naših čitalaca su uvek dobrodošle.

Osnovne povratne informacije: Pošaljite e-mail na adresu informatori@kombib.rs i u naslovu poruke napišite naslov knjige. Ako imate bilo kakva pitanja o bilo kom aspektu ove knjige, pošaljite nam e-mail na adresu informatori@kombib.rs.

Štamparske greške: Iako smo preduzeli sve mere da bismo obezbedili tačnost sadržaja, greške mogu da se potkradu. Ako pronađete grešku u ovoj knjizi, bili bismo zahvalni ako biste nam to prijavili. Posetite stranicu http://bit.ly/2H8K7oA, kliknite Ostavite komentar i unesite detalje.

Piraterija: Ako na Internetu pronađete ilegalnu kopiju naših knjiga, u bilo kojoj formi, molimo vas da nas o tome obavestite i pošaljete adresu lokacije ili naziv web sajta. Kontaktirajte sa nama na adresi informatori@kombib.rs i pošaljite nam link ka sumnjivom materijalu.

Ako ste zainteresovani da postanete autor: Ako postoji tema za koju ste specijalizovani i zainteresovani ste da pišete ili sarađujete na nekoj od knjiga, pogledajte vodič za autore na adresi www.packtpub.com/authors.

RECENZIJA

Kada pročitate i upotrebite ovu knjigu, zašto ne biste napisali vaše mišljenje na sajtu sa kojeg ste je poručili? Potencijalni čitaoci tada mogu da upotrebe vaše mišljenje da bi odlučili o kupovini, mi u „Packtu“ možemo da razumemo šta mislite o našim proizvodima, a naši autori mogu da vide povratne informacije o svojoj knjizi. Hvala!

1

Uvod u penetraciono testiranje i veb aplikacije

Veb aplikacija koristi HTTP protokol za klijent-server komunikaciju i zahteva veb pretraživač kao interfejs klijenta. To je verovatno najviše prisutan tip aplikacije u modernim kompanijama, od organizacionih anketa ljudskih resursa, do IT tehničkih servisa za veb sajt kompanije. Čak i aplikacije za mobilne uređaje i mnogi **Internet of Things (IoT)** uređaji koriste veb komponente kroz veb servise i veb interfejse koji su ugrađeni u njih.

Ne tako davno mislilo se da je bezbednost potrebna samo na perimetru organizacije i samo na nivou mreže, pa su kompanije trošile značajnu sumu novca za fizičku i mrežnu bezbednost. Međutim, to je samo dovelo do lažnog osećaja sigurnosti, zbog oslanjanja na veb tehnologije unutar i izvan organizacije. Poslednjih godina i meseci smo pojavile su se vesti o spektakularnom „curenju“ podataka i kršenju miliona zapisa, uključujući i informacije kao što su brojevi kreditnih kartica, istorije bolesti, kućne adrese i brojevi socijalnog osiguranja ljudi širom sveta. Mnogi od ovih napada su započeti eksploatisanjem ranjivosti Veba ili greške u dizajnu.

Moderne organizacije priznaju da zavise od veb aplikacija i veb tehnologija i da su podložne napadu kao i njihova mreža i operativni sistemi – ako ne i više. Rezultat toga su povećanje broja kompanija koje obezbeđuju zaštitu ili servise za odbranu od veb napada, pojava ili rast tehnologija, kao što su **Web Application Firewall (WAF)** i **Runtime Application Self-Protection (RASP)**, skeneri ranjivosti Veba i skeneri izvornog koda. Osim toga, povećan je u broj organizacija koje smatraju korisnim testiranje bezbednosti svojih aplikacija pre nego što ih izdaju krajnjim korisnicima, čime obezbeđuju mogućnost za talentovane hakere i profesionalce na polju bezbednosti da upotrebe svoje veštine za pronalaženje nepravilnosti i obezbede savet kako da se one isprave, pomažući time kompanijama, bolnicama, školama i vladama da imaju bezbednije aplikacije i obogate praksu razvoja softvera.

PROAKTIVNO TESTIRANJE BEZBEDNOSTI

Penetraciono testiranje i etičko hakovanje su proaktivni načini testiranja veb aplikacija izvršavanjem napada koji su slični stvarnim napadima, a mogu da se dese bilo kada. Ovo testiranje se vrši na kontrolisani način, u cilju pronalaženja što više nedostataka u bezbednosti i obezbeđivanja povratnih informacija kako da se izbegnu rizici koje izazivaju ovi nedostaci.

Veoma je korisno za kompanije da vrše testiranje bezbednosti u aplikacijama pre nego što ih izdaju krajnjim korisnicima. U stvari, postoje korporacije koje su svesne važnosti bezbednosti, pa su skoro u potpunosti integrisale penetraciono testiranje, procenu ranjivosti i preglede izvornog koda u svom ciklusu razvoja softvera. Prema tome, kada ove korporacije izdaju novu aplikaciju, ona je već prošla kroz različite faze testiranja i ispravki.

Različite metodologije testiranja

Neke ljude često zbunjuju sledeći termini - koriste ih naizmenično, ne razumevajući ih u potpunosti (iako se neki aspekti ovih termina preklapaju, postoji blaga razlika koja zahteva pažnju):

- etičko hakovanje
- penetraciono testiranje
- procena ranjivosti
- provere bezbednosti

Etičko hakovanje

Veoma malo ljudi shvata da je hakovanje pogrešno shvaćen termin; često se o hakeru razmišlja kao o zlonameranoj osobi koja sedi u mraku i nema društvenog života. Reč etički je ovde prefiks termina hakovanje. Termin etički haker se koristi za profesionalca koji radi na identifikaciji „rupa“ i ranjivosti u sistemima, o tome obaveštava prodavca ili vlasnika sistema i, ponekad, pomaže im da isprave greške u sistemu. Alatke i tehnike koje koriste **etički hakeri** su slične onima kojime se služe provalnik ili zlonamerni haker, ali je cilj drugačiji, jer se koriste na mnogo profesionalniji način. Etički hakeri su poznati i kao istraživači bezbednosti.

Penetraciono testiranje

Penetraciono testiranje je termin koji ćemo upotrebljavati veoma često u ovoj knjizi. Reč je o podskupu etičkog hakovanja. To je profesionalniji termin koji se koristi za opis onoga što radi etički haker. Ako planirate karijeru u etičkom hakovanju ili testiranju bezbednosti, često ćete videti oglase za posao u kojima se traži izvršilac penetracionog testa. Iako je penetraciono testiranje podskup etičkog hakovanja, razlikuje se na više načina. To je mnogo efikasniji način za identifikaciju ranjivosti u sistemima i otkrivanju da li se ranjivost može eksploatisati ili ne. Penetraciono testiranje se reguliše ugovorom između izvršioca i vlasnika sistema koji treba da se testira. Potrebno je definisati oblast testa, radi identifikovanja sistema koji će biti testirani, i pravila angažovanja koja određuju na koji će način biti izvršeno testiranje.

Procena ranjivosti

Ponekad će organizacije želeti da identifikuju samo ranjivosti koje postoje u njihovim sistemima, bez eksploatisanja i dobijanja pristupa. Procena ranjivosti je mnogo opširnija od penetracionih testova. Krajnji rezultat **procene ranjivosti** je izveštaj u kojem se opisuju pronađene ranjivosti - najteže su izlistane na vrhu liste, a one koje predstavljaju manji rizik nalaze se niže u izveštaju. Ovaj izveštaj je koristan za klijente koji znaju da imaju bezbednosne probleme i koji treba da identifikuju i odrede prioritete najkritičnijih.

Provere bezbednosti

Provera bezbednosti je sistematska procedura koja se koristi za merenje stanja sistema u odnosu na unapred određeni skup standarda. Ovi standardi mogu da budu industrijska najbolja praksa ili interna kontrolna lista. Primarni cilj provere bezbednosti je merenje i slanje izveštaja o usklađenosti. Ako proveravate bezbednost veb servera, prvo treba da pregledate otvorene portove na serveru, štetne HTTP metode, kao što je `TRACE`, uključene opcije na serveru, upotrebljeni standard enkripcije i dužinu ključa.

RAZMATRANJA PRILIKOM IZVRŠAVANJA PENETRACIONOG TESTIRANJA

Kada planirate da izvršite penetraciono testiranje projekta, bez obzira da li to radite za klijenta kao profesionalni izvršilac penetracionog testa ili kao član internog bezbednosnog tima kompanije, postoje aspekti koje uvek treba razmotriti pre početka njegove primene.

Pravila angažovanja

Pravila angažovanja (RoE) je dokumenat u kojem je opisan način na koji će penetracioni test biti izveden. Pre nego što započnete penetracioni test, u RoE-u treba da budu jasno naznačene sledeće direktive:

- tip i oblast važenja testiranja
- detalji o kontaktu klijenta
- obaveštenja IT tima klijenta
- obrada poverljivih podataka
- obaveštenja o statusu i izveštaji

Tip i oblast važenja testiranja

Tip testiranja može da bude crna kutija, bela kutija ili srednja siva kutija, u zavisnosti od načina kako je izvršeno angažovanje i od količine informacija koje se dele sa timom koji testira aplikaciju.

Postoje akcije koje mogu i koje ne mogu da budu izvršene u svakom tipu testiranja. U testiranju **crne kutije** tim za testiranje radi sa tačke gledišta napadača koji je van organizacije, jer izvršilac penetracionog testa započinje rad „od nule“ i pokušava da identifikuje mrežnu mapu, implementirane mehanizme zaštite, veb sajtove, veb servise i tako dalje. Iako je ovaj pristup realističniji u simuliranju eksternog napadača, treba da imate na umu da takve informacije mogu lako da budu sakupljene iz javnih izvora ili da napadač može da bude nezado-

voljni radnik ili bivši zaposleni koji već poseduje te informacije. Prema tome, primena tipa testiranja crne kutije može podrazumevati uzaludno trošenje vremena i novca - na primer, ako je cilj interna aplikacija koju koriste samo zaposleni u organizaciji.

Za test **bele kutije** neophodno je da timu za testiranje budu obezbeđene sve dostupne informacije o ciljevima (ponekad je u njih uključen i izvorni kod aplikacija), pa je potrebno kraće vreme za izviđanje i skeniranje. Za test sive kutije neophodno je da tim za testiranje raspolaže delimičnim informacije, kao što su URL-ovi aplikacija, dokumentacija nivoa korisnika i/ili korisnički nalozi.

Testiranje **sive kutije** je posebno korisno kada testirate veb aplikacije, jer je glavni cilj da pronađete ranjivosti unutar same aplikacije, a ne na hosting serveru ili na mreži. Izvršioci penetracionog testa mogu da koriste korisničke naloge da bi usvojili tačku gledišta zlonamernog korisnika ili napadača koji je dobio pristup pomoću društvenog inženjeringa.

Kada određujete oblast važenja testiranja, klijent zajedno sa timom za testiranje treba da proceni koje su informacije „ranjive", a koje je potrebno zaštititi. Na osnovu toga odlučite koje aplikacije/mreže treba da budu testirane i sa kojim stepenom pristupa informacijama.

Detalji o kontaktu klijenta

Složićete se da, čak i kada preduzmemo sve potrebne mere predostrožnosti kada se vrši testiranje, postoje situacije kada testiranje može krenuti loše, kada računari "rade loše stvari". Ako imamo tačne informacije o kontaktu na strani klijenta, to može da bude veoma korisno. Penetraciono testiranje se često pretvara u **Denial-of-Service (DoS)** napad. Tehnički tim na strani klijenta treba da bude dostupan u slučaju da dođe do pada sistema, pa je potrebno resetovanje za vraćanje na mrežu.

Penetraciono testiranje veb aplikacija ima prednosti zato što može da se izvrši u okruženju koje je specifično građeno za ovu namenu, omogućavajući izvršiocima testa da smanje rizik negativnog uticaja na produktivne elemente klijenta.

Obaveštenja IT tima klijenta

Penetracioni testovi se, takođe, koriste kao sredstvo za proveru spremnosti tima podrške u odgovaranju na incidente i pokušaje provale. Treba da razjasnite sa klijentom da li je to najavljeni ili nenajavljeni test. Ako je test najavljen, obavezno obavestite klijenta o vremenu, datumu i IP adresama sa kojih će testiranje (napad) biti obavljeno da biste izbegli da klijentov IT bezbednosni tim ne propusti neke stvarne pokušaje provale. Ako je test nenajavljen, dogovorite se sa klijentom šta će se desiti ako automatski sistem ili mrežni administrator blokiraju test. Da li će test biti završen ili ćete ga nastaviti? Sve zavisi od toga za šta je test namenjen - za proveru bezbednosti infrastrukture ili za proveru odgovora tima za mrežnu bezbednost i obradu incidenta. Čak i ako izvršavate nenajavljeni test, uverite se da neko u eskalacijskoj matrici zna vreme i datum izvršenja testa. Penetracioni testovi veb aplikacija su, obično, najavljeni.

Obrada poverljivih podataka

U toku pripreme i izvršenja testa timu za testiranje će biti obezbeđene i otkrivene poverljive informacije o kompaniji, njenom sistemu i/ili njenim korisnicima. Obradi poverljivih podataka treba posvetiti specijalnu pažnju u RoE-u i treba da se preduzmu pravilne mere skladištenja i komunikacije (na primer, potpuna enkripcija diska na računaru izvršioca testa, šifrirani izveštaji ako se oni šalju e-mailom i tako dalje). Ako klijent potpada pod jurisdikciju različitih regulacionih zakona, kao što su **Health Insurance Portability and Accountability Act (HIPAA)** i **Gramm-Leach-Bliley Act (GLBA)**, ili evropskih zakona o privatnosti podataka, samo autorizovanom osoblju može da se omogući da vidi lične podatke korisnika.

Obaveštenja o statusu i izveštaji

Komunikacija je ključ za uspešan penetracioni test. Redovni sastanci treba da budu održavani između organizacije klijenta i tima za testiranje, koji treba da obezbedi izveštaje o statusu. Tim za testiranje treba da predstavi koliko daleko je stigao u svom poslu i koje su ranjivosti je pronašao. Organizacija klijenta bi takođe trebalo da potvrdi da li su njeni sistemi za detekciju pokrenuli neka upozorenja kao rezultat pokušaja proboja. Ako je testiran veb server i upotrebljen WAF, pokušaji napada treba da budu evidentirani i blokirani. Najbolja praksa za tim za testiranje je da dokumentuje vreme kada je izvršen test. To će pomoći bezbednosnom timu da uporedi evidencije sa penetracionim testovima.

WAF-ovi funkcionišu analiziranjem HTTP/HTTPS saobraćaja između klijenata i servera i mogu da detektuju i blokiraju većinu uobičajenih napada na veb aplikacije.

Ograničenja penetracionog testiranja

Iako su penetracioni testovi preporučeni i treba da se izvode redovno, postoje određena ograničenja. Kvalitet testa i njegovi rezultati će direktno zavisiti od veštine tima za testiranje. Penetracionim testovima ne mogu da se pronađu sve ranjivosti zbog ograničenja oblasti važenja, ograničenja pristupa izvršiocima penetracionog testa za proveru okruženja i ograničenja u alatkama koje koriste izvršioci testiranja. Slede neka ograničenja penetracionog testa:

- **ograničenja veština** - Kao što je pomenuto ranije, uspeh i kvalitet testa će direktno zavisiti od veština i iskustva tima za penetraciono testiranje. Penetracioni testovi mogu da budu klasifikovani u tri kategorije: testiranje mreže, sistema i veb aplikacije. Nećete dobiti tačne rezultate ako osoba koja zna da izvrši penetraciono testiranje mreže radi na projektu koji uključuje testiranje veb aplikacije. Zbog ogromnog broja tehnologija koje su primenjene na Internetu, veoma je teško naći osobu koja ima potrebne veštine za izvođenje sve tri vrste testiranja. Izvršilac testa možda ima detaljno znanje o Apache veb serverima, ali možda će se prvi put susresti sa IIS serverom. Iskustvo takođe igra značajnu ulogu u uspešnosti testa; mapiranje manje rizičnih ranjivosti na sistemu koji ima visok nivo pretnji je veština koja se stiče samo iskustvom.
- **ograničenje u vremenu** - Penetraciono testiranje je često jednostavan projekat koji treba da bude završen u unapred određenom vremenu. Tim za testiranje treba da dobije rezultate i identifikuje ranjivosti u tom periodu. Sa druge strane, napadači imaju mnogo više vremena da rade na svojim napadima i mogu pažljivo da ih isplaniraju. Izvršioci penetracionog testa takođe treba da kreiraju izveštaj na kraju testiranja, opisujući metodologiju i identifikovane ranjivosti i dajući rezime. Snimci ekrana treba da se prave u regularnim intervalima, a zatim se dodaju u izveštaj. Naravno, napadač neće pisati izveštaje i može da posveti više vremena napadu.
- **ograničenje prilagođenih eksploatacija** - U nekim visokobezbednim okruženjima normalni radni okviri penetracionog testiranja i alatke se malo koriste i tim treba da razmišlja van okvira, kao što su kreiranje prilagođenih eksploatacija i ručno pisanje skriptova za postizanje cilja. Kreiranje eksploatacija je ekstremno dugotrajno i osetno utiče na budžet i vreme za test. U svakom slučaju, pisanje prilagođenih eksploatacija treba da bude deo portfolija svakog izvršioca penetracionog testa.

- **izbegavanje DoS napada** - Hakovanje i penetraciono testiranje predstavljaju umetnost primoravanja računara ili aplikacije da urade nešto za šta nisu namenjeni. Prema tome, povremeno test može da dovede do DoS napada, umesto da izvršilac testa dobije pristup sistemu. Mnogi izvršioci ne pokreću takve testove da bi izbegli nenamerno izazivanje pada sistema. Pošto sistemi nisu testirani na DoS napade, oni su podložniji napadima provalnika, koji samo traže takve sisteme kojima mogu da pristupe korišćenjem Interneta da bi stekli slavu, tako što će ih srušiti. Provalnici su nekvalifikovani pojedinci koji eksploatišu dobro poznate slabosti koje je jednostavno pronaći u računarskim sistemima da bi postali ozloglašeni i ne brinu o potencijalnim štetnim posledicama. Potrebno je podučavanje klijenata o vrlinama i manama DoS testa, jer će im ono pomoći da donesu odgovarajuće odluke.
- **ograničenje pristupa** - Mreže su podeljene na različite segmente i tim za testiranje će često imati pristup i pravo da testira samo one segmente koji imaju servere i koji su dostupni sa Interneta da bi bio simuliran stvarni napad. Međutim, takav test neće detektovati probleme konfiguracije i ranjivosti u internoj mreži gde se nalaze klijenti.
- **ograničenje upotrebljenih alatki** - Ponekad je timu za penetraciono testiranje dozvoljeno da koristi samo alatke i radne okvire eksploatacije koje je odobrio klijent. Ni jedna alatka nije kompletna, bez obzira da li je verzija besplatna ili komercijalna. Tim za testiranje treba da poznaje ove alatke i da ume da pronađe alternativu kada u njima nedostaju neke funkcije.

Da bi prevazišle ova ograničenja, velike organizacije imaju namenske timove za penetraciono testiranje koji pronalaze nove ranjivosti i regularno izvršavaju testove. Druge organizacije izvršavaju regularne preglede konfiguracije, pored penetracionih testova.

Potreba za testiranjem veb aplikacija

Zbog ogromnog broja veb sajtova na Internetu i povećanja broja organizacija koje svoje poslove obavljaju online, veb aplikacije i veb serveri su atraktivne mete za napadače. Veb aplikacije su svuda na javnim i privatnim mrežama, pa napadači ne treba da brinu o nedostatku meta. Za interakciju sa veb aplikacijom potreban je samo veb pretraživač. Neke od nedostataka u veb aplikacijama, kao što su greške u logici, može i laik da eksploatiše. Na primer, zbog loše implementacije logike, ako kompanija ima sajt za elektronsku trgovinu koji omogućava korisniku dodavanje proizvoda u korpu za kupovinu nakon procesa provere, a zlonamerni korisnik to otkrije prilikom pokušaja proboja i pronalaženja greške, moći će da iskoristi ovu akciju veoma jednostavno bez nekih specijalnih alatki.

Ranjivosti u veb aplikacijama takođe obezbeđuju sredstva za širenje zlonamernih programa i virusa, koji mogu da se raširе širom sveta u roku od nekoliko minuta. Sajber kriminalci ostvaruju znatne finansijske dobitke eksploatisanjem veb aplikacija i instaliranjem zlonamernih programa, koji će, zatim, biti prosleđeni korisnicima aplikacije.

Zaštitne barijere na ivici su popustljivije za ulazni HTTP saobraćaj ka veb serveru, pa napadaču nisu potrebni otvoreni specijalni portovi. HTTP protokol, koji je dizajniran pre mnogo godina, ne obezbeđuje nikakve ugrađene bezbednosne funkcije; to je tekstualni protokol i zahteva dodatno uslojavanje upotrebe HTTPS protokola da bi zaštitio komunikaciju. Takođe ne obezbeđuje identifikaciju individualnih sesija, što znači da programer treba da je dizajnira. Mnogi programeri su angažovani čim završe koledž i imaju samo teoretsko znanje o programskim jezicima, a ne i iskustvo u aspektima bezbednosti programiranja veb aplikacija. Čak i kada su programeri obavešteni o ranjivosti, potrebno im je duže vreme da je isprave, jer su zauzeti kreiranjem funkcija i poboljšanjima veb aplikacije.

Bezbedno kodiranje započinje fazom projektovanja veb aplikacija. Ova faza treba da bude integrisana rano u ciklusu razvoja. Kasnije integrisanje bezbednosti će biti veoma teško i zahteva mnogo prerade koda. Identifikovanje rizika i pretnji rano u fazi razvoja upotrebom modeliranja pretnji pomaže u minimiziranju ranjivosti u spremnom kodu veb aplikacije.

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

Razlozi za zaštitu od napada u veb aplikacijama

Neki od razloga zaštite od napada u veb aplikacijama su sledeći:

- zaštita podataka klijenata
- usklađenost sa zakonima i propisima
- gubitak reputacije
- gubitak prihoda
- zaštita od prekida poslovanja

Ako veb aplikacija vrši interakciju sa informacijama o kreditnim karticama i skladišti te informacije, treba da bude usklađena sa pravilima i propisima koje postavlja **Payment Card Industry (PCI)**. PCI ima specifične smernice, kao što je pregled koda zbog ranjivosti u veb aplikaciji ili instaliranje WAF-a za izbegavanje rizika.

Kada veb aplikacija nije testirana za ranjivosti i napadač dobije pristup podacima klijenta, može da utiče na brend kompanije ako klijent preda tužbu protiv kompanije zbog neadekvatne zaštite podataka. To može da dovede i do gubitka prihoda, jer će se mnogi klijenti prebaciti kod konkurentske kompanije koja će im osigurati bolju bezbednost podataka.

Napadi na veb aplikacije takođe mogu da rezultiraju prekidom usluga ako se radi o DoS napadu, ukoliko je server isključen sa mreže zbog brisanja otkrivenih podataka ili zbog forenzičkog ispitivanja. To može negativno da utiče na finansijske izveštaje.

Ovi razlozi treba da budu dovoljni da ubede više rukovodstvo organizacije da uloži dodatne resurse (novac, ljudstvo i veštine) da bi bila poboljšana bezbednost veb aplikacija.

KALI LINUX

U ovoj knjizi ćemo upotrebiti alatke koje obezbeđuje Kali Linux za izvršenje testiranja. Kali Linux je GNU/Linux distribucija, zasnovana na Debianu. Koriste ga profesionalci za bezbednost za izvršenje ofanzivnih bezbednosnih zadataka, a održava ga kompanija „Offensive Security“. Prethodnik Kali Linuxa je BackTrack - to je bila jedna od glavnih alatki koje su koristili izvršioci penetracionog testiranja u periodu dužem od šest godina (do 2013. godine), nakon čega je zamenjena Kali Linuxom. U avgustu 2015. godine izdata je druga verzija Kali Linuxa, pod nazivom koda Kali Sana, a u januaru 2016. godine ona je prebačena u radnu verziju.

To znači da je softver ažuriran bez prekida, bez potrebe za menjanjem verzije operativnog sistema. Kali Linux ima veliki skup popularnih alatki za hakovanje, koje su spremne za upotrebu u svim instaliranim zavisnostima. Detaljno ćemo opisati ove alatke i upotrebićemo ih za testiranje veb aplikacija koje su ranjive na glavne nedostatke koji se nalaze u veb aplikacijama iz stvarnog sveta.

PREGLED VEB APLIKACIJE ZA IZVRŠIOCE PENETRACIONOG TESTA

Veb aplikacije uključuju mnogo više od samog HTML koda i veb servera. Ako niste programer koji je aktivno uključen u razvoj veb aplikacije, verovatno ne poznajete interne poslove HTTP protokola i različite načine na koje veb aplikacija vrši interakciju sa bazom podataka i ne znate šta se tačno dešava kada korisnik klikne na link ili unese URL veb sajta u veb pretraživač.

Veoma je važno da kao izvršilac penetracionog testa razumete kako informacije teku od klijenta do servera i baze podataka, a zatim nazad do klijenta. U ovom odeljku ćemo uključiti informacije koje će pomoći pojedincima koji nemaju prethodno znanje o penetracionom testiranju veb aplikacije da iskoriste alatke koje su obezbeđene u Kali Linuxu za sprovođenje penetracionog testa Veba sa kraja na kraj. Detaljno ćemo pregledati sledeće:

- HTTP protokol
- zaglavlja u HTTP-u
- praćenje sesije pomoću „kolačića“
- HTML
- arhitekturu veb aplikacija

HTTP protokol

Osnovni protokol koji prenosi saobraćaj veb aplikacije između veb servera i klijenta je poznat kao **Hypertext Transport Protocol (HTTP)**. Najčešće upotrebljavana implementacija protokola HTTP/1.1 je definisana u RFC-ovima 7230-7237, koji su zamenili stariju verziju definisanu u RFC-u 2616. Najnovija verzija, poznata kao HTTP/2, izdata je u maju 2015. godine i definisana je u RFC-u 7540. Prvo izdanje HTTP/1.0 se sada smatra zastarelim i nije preporučljivo.

Kako se Internet razvijao, dodate su nove funkcije u novija izdanja HTTP protokola. U verziji HTTP/1.1 u način na koji HTTP podržava keširanje dodate su nove funkcije, kao što su trajne veze, metod `OPTIONS` i nekoliko drugih poboljšanja.

RFC je detaljni tehnički dokumenat u kome su opisani internet standardi i protokoli koje je kreirao Internet Engineering Task Force (IETF). Finalna verzija RFC dokumenta postaje standard koji može da se prati kada se implementira protokol u aplikacije.

HTTP je klijent-server protokol u kojem klijent (veb pretraživač) kreira zahtev za server, a server vraća odgovor na zahtev. Odgovor servera je, uglavnom, u formi stranica formatiranih kao HTML. Prema standardnom podešavanju, HTTP protokol koristi port 80, ali veb server i klijent mogu da budu konfigurisani za upotrebu drugog porta.

HTTP je tekstualni protokol, što znači da sve informacije između klijenta i servera „putuju" nešifrirane i svaki prosečni korisnik u lancu komunikacije može da ih vidi i razume. Za otklanjanje ovog nedostatka u dizajnu izdata je nova implementacija, koja uspostavlja kanal šifiririrane komunikacije sa **Secure Sockets Layer (SSL)** protokolom, a zatim kroz njega šalje HTTP pakete. To se zove HTTPS ili HTTP preko SSL-a. Poslednjih godina SSL protokol je sve češće zamenjivan novijim protokolom pod nazivom **Transport Layer Security (TLS)**, koji je trenutno u verziji 1.2.

Poznavanje HTTP zahteva i odgovora

HTTP zahtev je poruka koju klijent šalje serveru da bi dobio neke informacije ili izvršio neku akciju. HTTP zahtev ima dva dela koja su razdvojena praznom linijom: zaglavlje i telo. Zaglavlje sadrži sve informacije koje se odnose na sam zahtev, očekivani odgovor, „kolačiće" i druge relevantne kontrolne informacije, a telo sadrži razmenjene podatke. HTTP odgovor ima istu strukturu i menja sadržaj i upotrebu informacija koje sadrži.

Zaglavlje zahteva

Ovde je prikazan HTTP zahtev snimljen pomoću posrednika veb aplikacije kada se pretražuje www.bing.com.

```
GET / HTTP/1.1
Host: www.bing.com
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/60.0.3112.113 Chrome/60.0.3112.113 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en;q=0.6
Cookie: SRCHD=AF=NOFORM; SRCHUID=V=2&GUID=30674151A8BF404A8615B1B06E9FFC79&dmnchg=1; SRCHUSR=DOB=20170910;
_EDGE_S=F=1&SID=278218376F3962692F0512CE6EBF63EC; _EDGE_V=1; MUID=27E4EF9EFB9463C01439E567FA126225; MUIDB=27E4EF9EFB9463C01439E567FA126225;
SRCHHPGUSR=CW=1367&CH=626&DPR=1&UTC=600&WTS=63640613243; _SS=SID=278218376F3962692F0512CE6EBF63EC&bIm=0864436HV=1505016460
Connection: close
```

Prva linija u ovom zaglavlju ukazuje na metod zahteva: `GET`, zatraženi resurs: / (odnosno, osnovni direktorijum) i verziju protokola: HTTP 1.1. Postoji još nekoliko polja koja se mogu nalaziti u HTTP zaglavlju - opisaćemo najrelevantnija:

- **Host** - Specifikuje host i broj porta resursa koji je zatražen. Veb server možda sadrži više od jednog sajta ili može da sadrži tehnologije, kao što su deljeno hostovanje ili raspoređivanje opterećenja. Ovaj parametar se koristi za razlikovanje različitih sajtova/aplikacija koje služi ista infrastruktura.
- **User-Agent** - Ovo polje koristi server za identifikovanje tipa klijenta (odnosno, veb pretraživača) koji će primiti informaciju. Korisno je za programere, jer odgovor može da bude prilagođen u skladu sa konfiguracijom korisnika, pošto nisu sve funkcije u HTTP protokolu i u jezicima za veb razvoj kompatibilne sa svim pretraživačima.
- **Cookies** - „Kolačići“ su privremene vrednosti koje su razmenjene između klijenta i servera i upotrebljene, između ostalog, za čuvanje informacije o sesiji.
- **Content-Type** - Ukazuje serveru tip medija fajlova sadržanih unutar tela zahteva.
- **Authorization** - Pomoću ovog parametra HTTP omogućava proveru identiteta klijenta po zahtevu. Postoji više režima za proveru identiteta; najčešće upotrebljavani su Basic, Digest, NTLM i Bearer.

Zaglavlje odgovora

Prilikom primanja zahteva i obrade njegovog sadržaja server može da odgovori porukom, kao na sledećoj slici.

```
HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Length: 109264
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
P3P: CP="NON UNI COM NAV STA LOC CURa DEVa PSAa PSDa OUR IND"
Set-Cookie: SRCHD=AF=NOFORM; domain=.bing.com; expires=Tue, 10-Sep-2019 04:07:23 GMT; path=/
Set-Cookie: SRCHUID=V=2&GUID=30674151A8BF404A8615B1B06E9FFC79&dmnchg=1; domain=.bing.com; expires=Tue, 10-Sep-2019 04:07:23 GMT; path=/
Set-Cookie: SRCHUSR=DOB=20170910; domain=.bing.com; expires=Tue, 10-Sep-2019 04:07:23 GMT; path=/
Set-Cookie: _SS=SID=278218376F3962692F0512CE6EBF63EC; domain=.bing.com; path=/
X-MSEdge-Ref: Ref A: F9F5FFD9AFE145B98F3E98E03003E30D Ref B: SYDEDGE0412 Ref C: 2017-09-10T04:07:23Z
Set-Cookie: _EDGE_S=F=1&SID=278218376F3962692F0512CE6EBF63EC; path=/; httponly; domain=bing.com
Set-Cookie: _EDGE_V=1; path=/; httponly; expires=Fri, 05-Oct-2018 04:07:23 GMT; domain=bing.com
Set-Cookie: MUID=27E4EF9EFB9463C01439E567FA126225; path=/; expires=Fri, 05-Oct-2018 04:07:23 GMT; domain=bing.com
Set-Cookie: MUIDB=27E4EF9EFB9463C01439E567FA126225; path=/; httponly; expires=Fri, 05-Oct-2018 04:07:23 GMT
Date: Sun, 10 Sep 2017 04:07:23 GMT
Connection: close

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html lang="es"
```

Prva linija zaglavlja odgovora sadrži statusni kod (200), koji je trocifreni kod. Ovaj kod pomaže pretraživaču da razume status operacije. Slede detalji za nekoliko važnih polja:

- **Status code** - Ne postoji polje pod nazivom statusni kod, ali je vrednost prosleđena u zaglavlje. 2xx serija statusnih kodova se koristi za prosleđivanje uspešne operacije nazad u veb pretraživač. 3xx serija se koristi za ukazivanje na preusmeravanje kada server želi da se klijent poveže na drugi URL ako je veb stranica pomerena. 4xx serija se koristi za ukazivanje da postoji greška u zahtevu klijenta i da korisnik treba da modifikuje zahtev pre nego što ga ponovo pošalje. 5xx serija ukazuje na grešku na strani servera, jer server ne može da izvrši operaciju. U prethodnom zaglavlju statusni kod je 200, što znači da je operacija uspešno izvršena. Kompletnu listu HTTP statusnih kodova možete pronaći na adresi https://developer.mozilla.org/en-US/docs/Web/HTTP/Status.
- **Set-Cookie** - Ovo polje, ako je definisano, uspostaviće vrednost „kolačića“ u klijentu, koju server može da upotrebi za identifikovanje klijenta i skladištenje privremenih podataka.
- **Cache-Control** - Ukazuje da li sadržaji odgovora (slike, kod skripta ili HTML) treba da budu uskladišteni u kešu pretraživača ili ne, radi skraćivanja vremena učitavanja stranice i pokazuje način na koji bi to trebalo da se uradi.
- **Server** - Ovo polje ukazuje na tip i verziju servera. Pošto je ova informacija interesantna za potencijalne napadače, dobra praksa je da konfigurišete servere tako da izostave ovaj odgovor, kao što je slučaj i u zaglavlju prikazanom na prethodnoj slici.

- **Content-Length** - Ovo polje će sadržati vrednost koja ukazuje na broj bajtova u telu odgovora. Ta vrednost se koristi da bi drugi korisnik znao kada je završen aktuelni zahtev/odgovor.

Obimnu listu svih polja zaglavlja i njihove upotrebe možete da pronađete na URL-u http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html.

HTTP metodi

Kada klijent pošalje zahtev na server, takođe bi trebalo da informiše server koje akcije treba da budu izvršene u željenom resursu. Na primer, ako korisnik želi samo da pregleda sadržaje veb stranice, pozvaće metod `GET`, koji informiše servere da pošalju sadržaje veb stranice u veb pretraživač.

U ovom odeljku opisano je nekoliko metoda. Oni su interesantni za izvršioca penetracionog testa, jer ukazuju koji tip razmene podataka se dešava između dve krajnje tačke.

Metod GET

Metod `GET` se koristi za preuzimanje bilo kojih informacija koje je URL identifikovao ili koje je generisao proces identifikovanja informacija. `GET` zahtev može da koristi parametre od klijenta, koje su prosleđene u veb aplikaciju pomoću samog URL-a dodavanjem upitnika (?), iza kojeg se nalaze nazivi i vrednosti parametra. Kao što je prikazano u sledećem zaglavlju, kada šaljete upit za pretragu za web penetration testing u Bing pretraživaču, informacija je poslata pomoću URL-a.

Metod POST

Metod `POST` je sličan metodu `GET`. Koristi se za preuzimanje podataka sa servera, ali prosleđuje sadržaj pomoću tela zahteva. Pošto su podaci sada prosleđeni u telo zahteva, postaje mnogo teže napadaču da detektuje i napadne osnovnu operaciju. Kao što je prikazano u sledećem `POST` zahtevu, korisničko ime (`login`) i lozinka (`pwd`) nisu poslati u URL, već u telo koje je odvojeno od zaglavlja praznom linijom.

```
POST /shepherd/login HTTP/1.1
Host: 192.168.56.101
Content-Length: 34
Cache-Control: max-age=0
Origin: http://192.168.56.101
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/60.0.3112.113 Chrome/60.0.3112.113 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://192.168.56.101/shepherd/login.jsp
Accept-Language: es-ES,es;q=0.8,en;q=0.6
Cookie: PHPSESSID=pk6bniclk6ojock4igcojfcol1; Server=b3dhc3Bid2E=;
_railsgoat_session=BAh7B0kiD3Nlc3Npb25faWQGOgZFFkkiJTRlMGUwMTE1NZYyMmE3MmY1YThlMGQ4M2ZlZGY0OTBkBjsAVEkiEF9jc3JmX3Rva2VuBjsAFkkiMXdkTEZMdkVpSXJlWklTdGZ
XNk55ZXVJc3BndmMrSFVWaksraWJqNXNOVVU9BjsARg%3D%3D--1a8fc3db3a90bf4fb25d98ca98dd8a00c665f648; JSESSIONID=7FCC73610B721C133D756B050117C3C9;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close

login=admin&pwd=admin&submit=Login
```

Metod HEAD

Metod `HEAD` je identičan sa metodom `GET`, osim što server ne uključuje telo poruke u odgovoru; odnosno, odgovor `HEAD` zahteva je samo zaglavlje odgovora na `GET` zahtev.

Metod TRACE

Kada se koristi metod `TRACE`, primajući server vraća nazad `TRACE` odgovor sa originalnom porukom zahteva u telu odgovora. Metod `TRACE` se koristi za identifikovanje svih alternacija zahteva posredničkim uređajima, kao što su posrednički serveri i zaštitne barijere. Neki posrednički serveri edituju HTTP zaglavlje kada su paketi prosleđeni pomoću njega, što može da se identifikuje pomoću metoda `TRACE`. Ovaj metod se koristi za testiranje, jer omogućava da se prati šta je primljeno na drugom kraju.

Metodi PUT i DELETE

Metodi `PUT` i `DELETE` su deo WebDAV-a, koji je ekstenzija HTTP protokola, a omogućava upravljanje dokumentima i fajlovima na veb serveru. Koriste ih programeri za slanje spremnih veb stranica na veb server. `PUT` se koristi za slanje podataka na server, a `DELETE` za uklanjanje podataka. U modernim aplikacijama ova dva metoda se, takođe, koriste u veb servisima za izvršavanje specifičnih operacija u bazi podataka. `PUT` se koristi za umetanje ili modifikaciju zapisa, a `DELETE` za brisanje, za isključivanje ili za sprečavanje budućeg čitanja delova informacija.

Metod OPTIONS

Metod `OPTIONS` se koristi za slanje upita serveru za opcije komunikacije koje su dostupne u traženom URL-u. U sledećem zaglavlju možete da vidite odgovor na zahtev `OPTIONS`.

```
HTTP/1.1 200 OK
Date: Sun, 10 Sep 2017 16:24:15 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with
Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14
OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
Allow: GET,HEAD,POST,OPTIONS,TRACE
Vary: Accept-Encoding
Content-Length: 0
Content-Type: text/html
```

Razumevanje rasporeda HTTP paketa je veoma važno, jer on sadrži korisne informacije i nekoliko polja može da se kontroliše sa kraja korisnika, dajući napadaču mogućnost da injektira zlonamerne podatke ili manipuliše određenim ponašanjem aplikacija.

Zadržavanje sesija u HTTP-u

HTTP je klijent-server protokol bez uspostavljenog stanja, gde klijent kreira zahtev, a server odgovara podacima. Sledeći zahtev koji dolazi se tretira kao potpuno novi zahtev, koji nije povezan sa prethodnim. Dizajn HTTP zahteva je takav da oni svi zavise jedan od drugog. Kada dodate stavku u korpu za kupovinu dok kupujete online, aplikaciji je potreban mehanizam za povezivanje stavki sa vašim nalogom. Svaka aplikacija može da koristi različit način za identifikovanje svake sesije.

Najčešće upotrebljavana tehnika za praćenje sesija je korišćenje ID-a sesije (identifikatora) koji podešava server. Čim se proveri identitet korisnika validnim korisničkim imenom i lozinkom, korisniku je dodeljen jedinstveni nasumični ID sesije. U svaki zahtev koji klijent pošalje uključen je i jedinstveni ID sesije za povezivanje zahteva sa korisnikom čiji je identitet proveren. ID može da se deli pomoću metoda `GET` ili `POST`. Kada koristite metod `GET`, ID sesije će postati deo URL-a, a kada koristite metod `POST`, ID se deli u telu HTTP poruke. Server održava tabelu za mapiranje korisničkih imena sa dodeljenim ID-om sesije. Najveća prednost dodele ID-a sesije je što, čak i ako je HTTP bez uspostavljenog stanja, korisnik ne treba da se proverava prilikom svakog zahteva; pretraživač će predstaviti ID sesije i server će ga prihvatiti.

ID sesije ima i nedostatke: svako ko dobije pristup ID-u sesije može da se lažno predstavlja kao korisnik, jer se od njega ne zahtevaju korisničko ime i lozinka. Štaviše, jačina ID-a sesije zavisi od stepena nasumičnosti koja je upotrebljena za njegovo generisanje, što može da pomogne u sprečavanju napada.

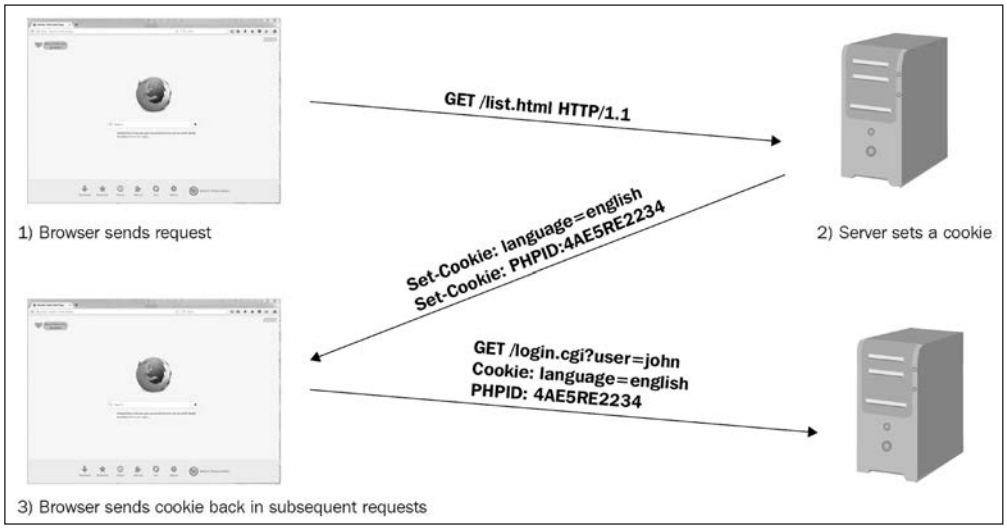
„Kolačići“

U HTTP komunikaciji „kolačić“ je deo informacije koji ima naziv, vrednost i parametre ponašanja, a skladišti ga server u fajl sistemu klijenta ili u memoriji veb pretraživača. „Kolačići“ su standardni mehanizam kroz koji se ID sesije prosleđuje između klijenta i veb servera. Kada ih koristite, server dodeljuje klijentu jedinstveni ID podešavanjem polja Set-Cookie u zaglavlju HTTP odgovora. Kada klijent primi zaglavlje, uskladištiće vrednost „kolačića“, odnosno uskladištiće ID sesije unutar lokalnog fajla ili memorije pretraživača i povezaće ga sa URL-om veb sajta koji ga je poslao. Kada korisnik ponovo poseti originalni veb sajt, pretraživač će poslati vrednost „kolačića“, identifikujući korisnika.

Osim za praćenje sesije, „kolačići“ mogu da se upotrebe i za skladištenje informacija za krajnjeg klijenta, kao što su jezik i druge konfiguracione opcije koje će biti trajne između sesija.

Tok „kolačića“ između servera i klijenta

„Kolačiće“ uvek postavlja i kontroliše server. Veb pretraživač je odgovoran samo za njihovo slanje do servera sa svakim zahtevom. Na sledećem dijagramu možete da vidite da je kreiran `GET` zahtev za server, a veb aplikacija na serveru podešava identifikaciju korisnika i jezik koji je korisnik izabrao u prethodnim zahtevima. U sledećim zahtevima koje klijent kreira „kolačić“ postaje deo zahteva:



Trajni i privremeni „kolačići“

„Kolačići“ su podeljeni u dve glavne kategorije - trajni i privremeni. Trajni su uskladišteni u internom skladištu uređaja klijenta kao tekstualni fajlovi. Pošto su „kolačići“ uskladišteni na hard drajvu, oni bi preživeli pad pretraživača ili bi postojali u različitim sesijama. Različiti pretraživači će na različite načine skladištiti trajne „kolačiće“. Na primer, Internet Explorer ih skladišti u tekstualnim fajlovima unutar direktorijuma korisnika AppData\Roaming\Microsoft\Windows\Cookie, a Google Chrome koristi SQLite3 bazu podataka koja se takođe nalazi u direktorijumu korisnika AppData\Local\Google\Chrome\User Data\Default\cookies. Kao što je prethodno pomenuto, „kolačići“ mogu da se upotrebe za praćenje poverljivih informacija u obliku ID-a sesije, parametara i podataka kupovine. Ako su sačuvani na hard drajvu, ne mogu da budu zaštićeni da ih ne modifikuje neki zlonamerni korisnik.

Da bi rešili bezbednosne probleme koji se odnose na trajne „kolačiće“, programeri su osmislili drugu vrstu „kolačića“ koji se u današnje vreme češće koriste, a poznati su kao **privremeni „kolačići“**, koji su uskladišteni u memoriji veb pretraživača, ne ostavljaju tragove na hard drajvu i prosleđuju se između veb pretraživača i servera pomoću zaglavlja zahteva i odgovora. Privremeni „kolačić“ je validan samo u određenom periodu, koji određuje server.

Parametri „kolačića“

Osim naziva i vrednosti, postoji i nekoliko drugih parametara koje podešava veb server, a koji definišu domet i dostupnost „kolačića“, kao što je prikazano u sledećem zaglavlju odgovora:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Date: Tue, 25 Nov 2014 18:22:25 GMT
Set-Cookie: ID=b34erdfWS; Domain=email.com; Path=/mail; Secure; HttpOnly; Expires=Wed, 26 Nov 2014 10:18:14 GMT
```

Slede detalji o nekim parametrima:

- **Domain** - Specifikuje domen kojem će „kolačić“ biti poslat.
- **Path** - Za dalje zaključavanje „kolačića“ ovaj parametar može da bude specifikovan. Ako je specifikovani domen email.com, a putanja je podešena na / mail, „kolačić“ će biti poslat samo na stranice unutar stranice email.com/mail.
- **HttpOnly** - Ovo je parametar koji je podešen za izbegavanje rizika koje predstavljaju Cross-site Scripting (XSS) napadi, jer JavaScript ne može da pristupi „kolačiću“.
- **Secure** - Ako je podešen, „kolačić“ mora da bude poslat samo preko sigurnih kanala komunikacije, konkretno SSL-a i TLS-a.
- **Expires** - „Kolačić“ će biti uskladišten do vremena specifikovanog u ovom parametru.

HTML podaci u HTTP odgovoru

Podaci u telu odgovora su informacije koje su korisne krajnjem korisniku. Ovo telo obično sadrži podatke formatirane kao HTML, ali to mogu da budu i **JavaScript Object Notation (JSON)** ili **eXtensible Markup Language (XML)** podaci, kod skripta ili binarni fajlovi, kao što su slike i video snimci. Prvobitno su samo tekstualne informacije bile originalno sačuvane na Vebu, formatirane na način koji je bio prikladniji za čitanje, ali su mogle da uključuju i tabele, slike i linkove ka drugim dokumentima - **Hypertext Markup Language (HTML)**, a veb pretraživač je bila alatka namenjena za interpretiranje HTML-a. HTML tekst je formatiran pomoću oznaka.

HTML nije programski jezik.

Kod na strani servera

Kod skripta i HTML formatiranje interpretira i predstavlja veb pretraživač. To se naziva kod na **strani klijenta**. Procesi koji su uključeni u preuzimanje informacija koje zahteva klijent, praćenje sesije i veći deo logike aplikacije izvršeni su na serveru pomoću koda na **strani servera**, napisanog jezikom, kao što su PHP, ASP.NET, Java, Python, Ruby i JSP. Ovaj kod kreira ispis, koji, zatim, može da bude formatiran pomoću HTML-a. Kada se URL završava ekstenzijom .php, znači da stranica može da sadrži PHP kod. Ta stranica mora da se pokrene kroz PHP mehanizam servera, koji omogućava da bude generisan dinamički sadržaj kada je učitana veb stranica.

Višeslojna veb aplikacija

Pošto se danas koriste složenije veb aplikacije, tradicionalna sredstva njihovog raspoređivanja na jedan sistem su zastarela. "Postavljanje svih jaja u jednu korpu" nije pametan način raspoređivanja poslovne aplikacije, jer to može da utiče na performansu, bezbednost i dostupnost aplikacije. Jednostavan dizajn jednog servera koji hostuje aplikaciju, kao i podatke, funkcioniše dobro za male veb aplikacije koje nemaju veliki saobraćaj. Troslojni metod dizajniranja veb aplikacije je mnogo bolji.

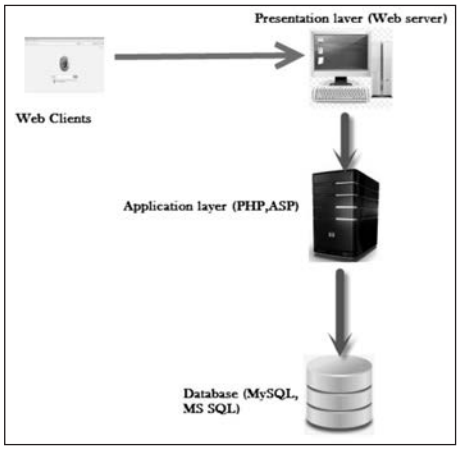
Dizajn troslojne veb aplikacije

U troslojnoj veb aplikaciji postoji fizička odvojenost između slojeva prezentacije, aplikacije i sloja podataka, što je i ovde opisano:

- **sloj prezentacije** - Ovo je server koji prima konekcije klijenta i predstavlja izlaznu tačku kroz koju je odgovor poslat nazad klijentu. Ovo je čeoni prikaz aplikacije. **Sloj prezentacije** je važan za veb aplikaciju, jer je to interfejs između korisnika i ostatka aplikacije. Podaci primljeni u sloj prezentacije su prosleđeni u komponente u sloju aplikacije za obradu. Primljeni rezultat je formatiran pomoću HTML-a i prikazan je na veb klijentu korisnika. Programi otvorenog kodaApache i nginx i komercijalni softver Microsoft IIS raspoređeni su u sloju prezentacije.
- **sloj aplikacije** - Procesorska intenzivna obrada i glavna logika aplikacije se čuvaju u **sloju aplikacije.** Kada sloj prezentacije sakupi potrebne podatke od klijenta i prosledi ih u sloj aplikacije, komponente koje funkcionišu na ovom sloju mogu da primene poslovnu logiku na podatke. Rezultat se vraća u sloj prezentacije i biće poslat nazad klijentu. Ako je klijent zatražio podatke, oni su ekstrahovani iz sloja podataka, obrađeni u korisnu formu za klijenta i prosleđeni u sloj prezentacije. Java, Python, PHP i ASP.NET su programski jezici koji funkcionišu u sloju aplikacije.

- **sloj pristupa podacima** - Stvarno skladište i skladište podataka funkcionišu na sloju pristupa podacima. Kada klijent zahteva podatke ili šalje podatke za skladištenje, sloj aplikacije prosleđuje podatke u sloj pristupa podacima za trajno skladištenje. Komponente koje funkcionišu na ovom sloju su odgovorne za održavanje podataka i čuvanje integriteta i dostupnosti. One su odgovorne i za upravljanje konkurentnim konekcijama iz sloja aplikacije. MySQL i Microsoft SQL su dve najčešće upotrebljavane tehnologije koje funkcionišu na ovom sloju. **Structured Query Language (SQL)** relacione baze podataka se najčešće koriste u veb aplikacijama, mada se NoSQL baze podataka, kao što su MongoDB, CouchDB i druge NoSQL baze podataka koje čuvaju informacije u obliku drugačijem od tradicionalnog formata tabele relacionih baza podataka, takođe često koriste, posebno u Big Data Analysis aplikacijama. SQL je definicija podataka i jezik za slanje upita koji podržavaju mnogi proizvodi baze podataka kao standard za preuzimanje i slanje podataka.

Na sledećem dijagramu prikazano je kako zajedno funkcionišu slojevi prezentacije, aplikacije i pristupa podacima.



Veb servisi

Veb servisi mogu da budu pregledani kao veb aplikacije koje ne uključuju sloj prezentacije. Arhitektura orijentisana ka servisu omogućava provajderu veb servisa da se lako integriše sa korisnikom određenog servisa. Veb servisi omogućavaju različitim aplikacijama da dele podatke i funkcionalnosti između sebe. Oni omogućavaju korisnicima da pristupe podacima preko Interneta, a aplikacija ne treba da zna format ili lokaciju podataka.

Ovo je veoma važno kada ne želite da otkrijete model podataka ili logiku koja je upotrebljena za pristup podacima, ali želite da podaci budu dostupni korisnicima. Dobar primer je veb servis koji otkriva berza. Online brokeri mogu da upotrebe ovaj veb servis da bi dobili informacije u realnom vremenu o deonicama i da bi ih prikazali na svojim veb sajtovima u njihovim stilu prezentovanja za krajnje kupce. Veb sajt brokera samo treba da pozove servis i zatraži podatke za kompaniju. Kada servis odgovori i pošalje podatke, veb aplikacija može da raščlani informacije i da ih prikaže.

Veb servisi su nezavisni od platforme. Aplikacija za berzu može da bude napisana bilo kojim jezikom, a servis će i dalje moći da se pozove, bez obzira na osnovnu tehnoloiju koja je upotrebljena za izgradnju aplikacije. Jedino o čemu treba da se slože provajder servisa i korisnik su pravila za razmenu podataka.

Trenutno postoji dva načina za razvoj veb servisa:

- **Simple Object Access Protocol (SOAP)**
- **Representational State Transfer (REST)**, takođe poznat kao RESTful veb servis

Predstavljanje SOAP i REST veb servisa

SOAP je tradicionalni metod za razvoj veb servisa, ali pošto ima mnogo nedostataka, aplikacije su sada prebačene na REST ili RESTful veb servise. XML je jedini dostupan format razmene podataka kada koristite SOAP veb servis, dok REST veb servisi mogu da koriste JSON i druge formate podataka. Iako su veb servisi koji su zasnovani na SOAP-u i dalje preporučljivi u nekim slučajevima zbog dodatnih bezbednosnih specifikacija, jednostavan REST veb servis je omiljeni metod mnogih programera veb servisa, zbog njegove jednostavnosti. SOAP je protokol, dok je REST arhitekturni stil. Amazon, Facebook, Google i Yahoo! su se već prebacili na REST veb servise.

Neke od karakteristika REST veb servisa su sledeće:

- Funkcionišu veoma dobro sa CRUD operacijama.
- Imaju bolju performansu i skalabilnost.
- Obrađuju više ulaznih i izlaznih formata.
- Jednostavniji su za učenje za programere koji se povezuju na veb servise.
- Filozofija REST dizajna je slična veb aplikacijama.

CRUD je skraćenica za create, read, update i delete (kreiranje, čitanje, ažuriranje i brisanje); opisuje četiri osnovne funkcije trajnog skladišta.

Glavna prednost SOAP-a u odnosu na REST veb servis je što je SOAP veb servis nezavisan za transport, dok REST funkcioniše samo pomoću HTTP-a. REST je zasnovan na HTTP-u i, prema tome, iste ranjivosti koje utiču na standardne veb aplikacije mogu da utiču i na REST servise. Srećom, ista bezbednosna najbolja praksa može da se primeni za obezbeđivanje REST veb servisa.

Složenost svojstvena razvoju SOAP servisa gde su XML podaci obuhvaćeni u SOAP zahtev, a zatim poslati pomoću HTTP-a, primorala je mnoge organizacije da se prebace na REST servise. Takođe je potreban i **Service Definition Language (WSDL)** fajl, koji obezbeđuje informacije koje se odnose na servis. Direktorijum UDDI treba da se održava i u njemu će biti publikovan WSDLfajl.

Osnovna ideja REST servisa je da, umesto da se koriste složeni mehanizmi, kao što je SOAP, REST servis direktno komunicira sa provajderom servisa pomoću HTTP-a, bez potrebe za dodatnim protokolom. Ovaj servis koristi HTTP za kreiranje, čitanje, ažuriranje i brisanje podataka.

Zahtev koji pošalje korisnik veb servisa zasnovanog na SOAP-u je sledeći:

```
    <?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:body sp="http://www.stockexchange.com/stockprice">
        <sp:GetStockPrice>
          <sp:Stockname>xyz</sp:Stockname>
        </sp:GetStockPrice>
      </soap:Body>
    </soap:Envelope>
```

Sa druge strane, zahtev poslat u REST veb servis može da bude jednostavan, kao što je sledeći:

```
http://www.stockexchange.com/stockprice/Stockname/xyz
```

Aplikacija koristi `GET` zahtev za čitanje podataka iz veb servisa, koji ima malu dodatnu memoriju i, za razliku od dugačkog i komplikovanog SOAP zahteva, veoma je jednostavan programerima za kodiranje. Iako REST veb servis takođe može da vrati podatke pomoću XML-a, JSON se češće koristi za vraćanje podataka.

HTTP metodi u veb servisima

REST veb servisi mogu da tretiraju HTTP metode drugačije nego u standardnoj veb aplikaciji. Ovo ponašanje zavisi od preferenci programera, ali je sve popularnije povezivanje `POST`, `GET`, `PUT` i `DELETE` metoda sa CRUD operacijama. Najčešće upotrebljavani pristup je sledeći:

- kreiranje: `POST`
- čitanje: `GET`
- ažuriranje: `PUT`
- brisanje: `DELETE`

Neke Application Programming Interface (API) implementacije zamenjuju `PUT` i `POST` funkcionalnosti.

XML i JSON

Formate XML i JSON koriste veb servisi za predstavljanje strukturiranih skupova podataka ili objekata.

Kao što je opisano u prethodnim odeljcima, XML koristi sintaksu koja je zasnovana na oznakama i svojstva i vrednosti za te oznake - na primer, meni **File** u aplikaciji može da bude predstavljen na sledeći način:

```
<menu id="m_file" value="File">
  <popup>
    <item value="New" onclick="CreateDocument()" />
    <item value="Open" onclick="OpenDocument()" />
    <item value="Close" onclick="CloseDocument()" />
  </popup>
</menu>
```

Nasuprot tome, JSON koristi ekonomičniju sintaksu, koja podseća na programske jezike C i Java. Isti meni u JSON formatu će biti sledeći:

```
{"menu": {
  "id": "m_file",
  "value": "File",
  "popup": {
    "item": [
      {"value": "New", "onclick": "NewDocument()"},
      {"value": "Open", "onclick": "OpenDocument()"},
      {"value": "Close", "onclick": "CloseDocument()"}
    ]
  }
}}
```

AJAX

Asynchronous JavaScript and XML (AJAX) je kombinacija više postojećih veb tehnologija, koja omogućava klijentu da pošalje zahteve i obrađuje odgovore u pozadini, bez direktne intervencije korisnika. Takođe omogućava da oslobodite server nekog dela zadataka u obradi logike aplikacije. AJAX omogućava da komunicirate sa veb serverom, bez potrebe da korisnik eksplicitno kreira novi zahtev u veb pretraživaču. Rezultat je brži odgovor sa servera, jer delovi veb stranice mogu da budu ažurirani posebno, a to obogaćuje korisničko iskustvo. AJAX koristi JavaScript za povezivanje i preuzimanje informacija sa servera, bez ponovnog učitavanja cele veb stranice.

Slede neke od prednosti upotrebe AJAX-a:

- **povećanje brzine** - Cilj upotrebe AJAX-a je poboljšanje performanse veb aplikacije. Ažuriranje pojedinačnih elemenata zahteva minimalnu obradu na serveru i stoga je poboljšana performansa. Odgovaranje na strani klijenta je takođe drastično poboljšano.
- **jednostavnost** - U aplikaciji zasnovanoj na AJAX-u korisnik ne treba ponovo da učita celu stranicu da bi „osvežio“ specifične delove veb sajta. To aplikaciju čini interaktivnijom i jednostavnijom, pa takođe može da se upotrebi za izvršenje validacije u realnom vremenu i automatsko dovršavanje unosa.
- **asinhroni pozivi** - Aplikacije zasnovane na AJAX-u su dizajnirane da kreiraju asinhrone pozive za veb server, pa otuda i naziv Asynchronous JavaScript and XML. To omogućava korisniku da vrši interakciju sa veb stranicom dok se jedan njen deo ažurira „iza scene“.

- **smanjena upotreba mreže** - Neizvršavanjem potpunog „osvežavanja“ stranice svaki put smanjena je i upotreba mreže. U veb aplikacijama u kojima postoje velike slike, video snimci ili dinamički sadržaj, kao što su Java apleti, ili se učitavaju Adobe Flash programi, upotreba AJAX-a može da optimizuje upotrebu mreže.

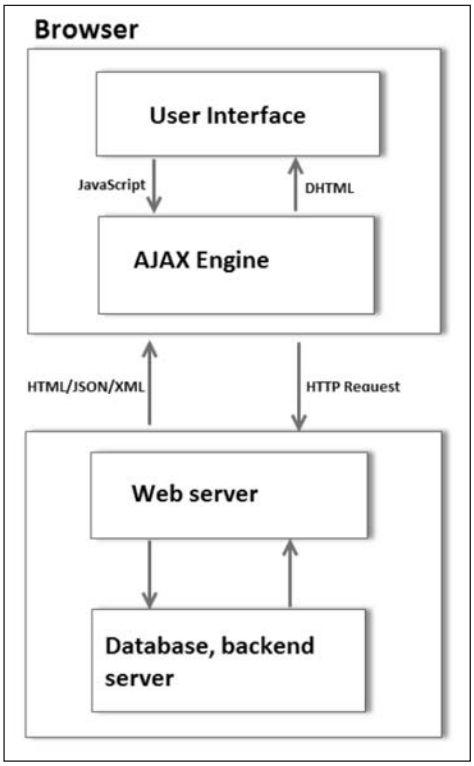
Gradivni blokovi AJAX-a

Kao što je ranije pomenuto, AJAX je kombinacija uobičajenih veb tehnologija koje se koriste za izgradnju veb aplikacije. Način na koji je aplikacija dizajnirana pomoću ovih veb tehnologija rezultira aplikacijom zasnovanom na AJAX-u. Slede neke komponente AJAX-a:

- **JavaScript** - Najvažnija komponenta aplikacije zasnovane na AJAX-u je JavaScript kod na strani klijenta. JavaScript vrši interakciju sa veb serverom u pozadini i obrađuje informacije pre nego što su one prikazane korisniku. JavaScript koristi **XMLHttpRequest (XHR)** API za prenos podataka između servera i klijenta. XHR postoji u pozadini i korisnik nije svestan njegovog postojanja.
- **dinamički HTML (DHTML)** - Kada su podaci preuzeti sa servera i JavaScript ih obradi, elementi veb stranice treba da budu ažurirani da bi reflektovali odgovor sa servera. Odličan primer je kada unosite korisničko ime prilikom popunjavanja online obrasca. Obrazac se dinamički ažurira da bi reflektovao i informisao korisnika da li je korisničko ime već registrovano na veb sajtu. Koristeći DHTML i JavaScript, možete da ažurirate sadržaj stranice u toku rada. DHTML je postojao znatno pre AJAX-a. Glavni nedostatak upotrebe samo DHTML-a je što on zavisi od koda na strani klijenta za ažuriranje stranice. Uglavnom, ne treba da imate ništa učitano na strani klijenta i treba da vršite interakciju sa kodom na strani servera. Tada se u igru uključuje AJAX koji kreira konekciju između koda na strani klijenta i koda na strani servera pomoću XHR objekata. Pre AJAX-a treba da koristite JavaScript aplete.
- **Document Object Model (DOM)** - DOM je radni okvir upotrebljen za organizovanje elemenata u HTML ili XML dokumentu. To je konvencija za predstavljanje i interakciju sa HTML objektima. Zamislite da je HTML dokumenat raščlanjen kao stablo, na kojem je svaki elemenat prikazan kao čvor stabla, a svaki čvor stabla ima svoje atribute i događaje. Na primer, objekat tela HTML dokumenta će imati specifičan skup atributa, kao što su text, link, bgcolor i tako dalje. Svaki objekat takođe ima događaje. Ovaj model omogućava interfejsu za JavaScript da pristupi i dinamički ažurira sadržaje na stranici pomoću DHTML-a. DHTML je funkcija pretraživača, a DOM se ponaša kao interfejs za njenu upotrebu.

Tok rada AJAX-a

Na sledećoj slici ilustrovana je interakcija između različitih komponenata aplikacije zasnovane na AJAX-u. Kada se uporedi sa tradicionalnom veb aplikacijom, AJAX mehanizam je glavni dodatak. Dodatak sloja AJAX mehanizma se ponaša kao posrednik za sve zahteve i odgovore kreirane kroz AJAX. AJAX mehanizam je JavaScript interpreter.



Sledi prikaz toka rada korisnika koji vrši interakciju sa aplikacijom zasnovanom na AJAX-u (korisnički interfejs i AJAX mehanizam su komponente u veb pretraživaču klijenta):

1. Korisnik unosi URL veb stranice, a pretraživač šalje HTTP zahtev serveru. Server obrađuje zahtev i šalje nazad odgovor sa HTML sadržajem, koji je prikazan u pretraživaču kroz mehanizam za renderovanje Veba. U HTML-u veb stranica je ugrađena u JavaScript kod koji izvršava JavaScript interpreter kada se desi događaj.
2. Kada vrši interakciju sa veb stranicom, korisnik se susreće sa elementom koji koristi ugrađeni JavaScript kod i pokreće događaj. Primer za to je stranica Google pretrage. Čim korisnik počne da unosi upit za pretragu, pozadinski AJAX mehanizam presreće zahtev korisnika. AJAX mehanizam prosleđuje zahtev na

server pomoću HTTP zahteva. Ovaj zahtev je transparentan korisniku i korisnik ne treba da klikne na dugme za podnošenje zahteva ili da „osvežava“ celu stranicu.

3. Na strani servera sloj aplikacije obrađuje zahtev i vraća podatke nazad u AJAX mehanizam u JSON, HTML ili XML formatu. AJAX mehanizam prosleđuje ove podatke u mehanizam za renderovanje Veba i oni će biti prikazani u veb pretraživaču. Veb pretraživač koristi DHTML za ažuriranje samo selektovanog odeljka veb stranice da bi reflektovao nove podatke.

Ne zaboravite sledeće dodatne stavke kada se susretnete sa aplikacijom zasnovanom na AJAX-u:

- XMLHttpRequest API čini „iza scene“ svoju „magiju“. Obično se naziva samo XHR, zbog dugačkog naziva. JavaScript objekat pod nazivom xmlhttp je prvo instanciran i upotrebljen je za slanje i preuzimanje odgovora sa servera. Podrška pretraživača za XHR je potrebna da bi AJAX funkcionisao. Sve novije verzije vodećih veb pretraživača podržavaju ovaj API.
- XML deo AJAX-a je malo zbunjujući. Aplikacija može da upotrebi bilo koji format, osim XML-a, kao što su JSON, tekstualni format, HTTP ili čak i slike, kada razmenjuje podatke između AJAX mehanizma i veb servera. JSON je omiljeni format, jer je lak i može da se pretvori u JavaScript objekat, koji dalje omogućava skriptu da pristupi podacima i manipuliše njima.
- Višestruki asinhroni zahtevi mogu da se dese istovremeno, bez čekanja da se jedan zahtev završi.
- Mnogi programeri koriste AJAX radne okvire, koji pojednostavljuju dizajniranja aplikacije. JQuery, Dojo Toolkit, **Google Web Toolkit (GWT)** i Microsoft AJAX library (.NET aplikacije) su dobro poznati radni okviri.

Primer za AJAX zahtev je sledeći:

```
function loadfile()
{
  //initiating the XMLHttpRequest object
  var xmlhttp;
  xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange=function()
  {
    if (xmlHttp.readyState==4)
    {
      showContents(xmlhttp.ResponseText);
    }
  //GET method to get the links.txt file
  xmlHttp.open("GET", "links.txt", true);
```

Funkcija `loadfile()` prvo instancira `xmlhttp` objekat, a zatim ga koristi za povlačenje tekstualnog fajla sa servera. Kada server vrati tekstualni fajl, prikazuju se sadržaji fajla. Fajl i njegovi sadržaji su učitani bez uključivanja korisnika, kao što je prikazano u prethodnom isečku koda.

HTML5

Peta verzija HTML specifikacije je prvi put publikovana u oktobru 2014. godine. Ova nova verzija specifikuje API-je za reprodukovanje medija, prevlačenje i otpuštanje, veb skladištenje, editabilni sadržaj, geolokaciju, lokalne SQL baze podataka, kriptografiju, veb utičnice i mnoge druge, što može biti interesantno sa stanovišta testiranja bezbednosti, jer su otvorene nove putanje za napade ili pokušaje za rešavanje bezbednosnih problema u prethodnim verzijama HTML-a.

WebSocket

HTTP je protokol bez uspostavljenog stanja, kao što je ranije pomenuto. To znači da je nova konekcija uspostavljena za svaki zahtev i zatvorena nakon svakog odgovora. HTML5 WebSocket je interfejs za komunikaciju koji omogućava trajnu dvosmernu konekciju između klijenta i servera.

WebSocket otvara klijent pomoću `GET` zahteva, kao što je sledeći:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin:  http://example.com
```

Ako server razume zahtev i prihvati konekciju, njegov odgovor će biti sledeći:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

HTTP konekcija je zamenjena WebSocket konekcijom i postaje dvosmerni binarni protokol, koji ne mora da bude kompatibilan sa HTTP-om.

REZIME

Ovo poglavlje je bilo uvod u etičko hakovanje i penetraciono testiranje veb aplikacija. Započeli smo ga identifikovanjem različitih načina testiranja veb aplikacija. Takođe smo opisali važna pravila angažovanja koja treba da budu definisana pre pokretanja testa. Zatim smo ispitali važnost testiranja veb aplikacija u današnjem svetu i rizike ako se ne izvrši regularno testiranje. Ukratko smo predstavili Kali Linux kao platformu za testiranje i napravili kratak pregled koncepata i tehnologija koje koriste moderne veb aplikacije.