SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Računalna forenzika - Seminar

HTTP Protokol

Mate Šimović

Zagreb, siječanj 2018

SADRŽAJ

1. Uvod

HTTP protokol je osnovica *World Wide Weba* i stoga jedan od najkorištenijih telekomunikacijskih protokola današnjice.
Svatko tko posjećuje WEB stranice, internetske portale, društvene mreže ili umrežene mobilne aplikacije najčešće pri tome koristi HTTP protokol. Također, svaki razvojni programer koji razvija WEB aplikacije, te svaki WEB poslužitelj koristi HTTP protokol.

Pri tome, većina prosječnih korisnika WEB-a ima povjerenje u sigurnost i zaštitu HTTP protokola te očekuje da su pri takvim aktivnostima njihovi osobni podaci, lozinke i osjetljivi podaci u dobroj mjeri zaštićeni, te da se zapisi o njihovoj aktivnost na internetu mogu isključivo pronaći u povijesti WEB preglednika na njihovom privatnom računalu.
S druge strane, čak se i dobar dio razvojnih programera, premda više upućen u sigurnosne probleme na internetu ne bavi direktno pitanjima sigurnosti, nego vjeruje u sigurnost HTTP protokola ili brigu o sigurnosti svojih aplikacija prepušta razvojnom okviru (eng. *framework*) kojega koristi.

Ipak, unatoč velikoj raširenosti HTTP protokola i tome što je velika količina truda uložena u razvoj sigurnosti HTTP-a, postoje ranjivosti u protokolu koje se mogu zloupotrijebiti, te postoje različiti mehanizmi izvođenja napada na sustave koji koriste HTTP protokol, što može dovesti do zloupotrebe osobnih podataka korisnika i do štete na WEB aplikacijama i sustavima.

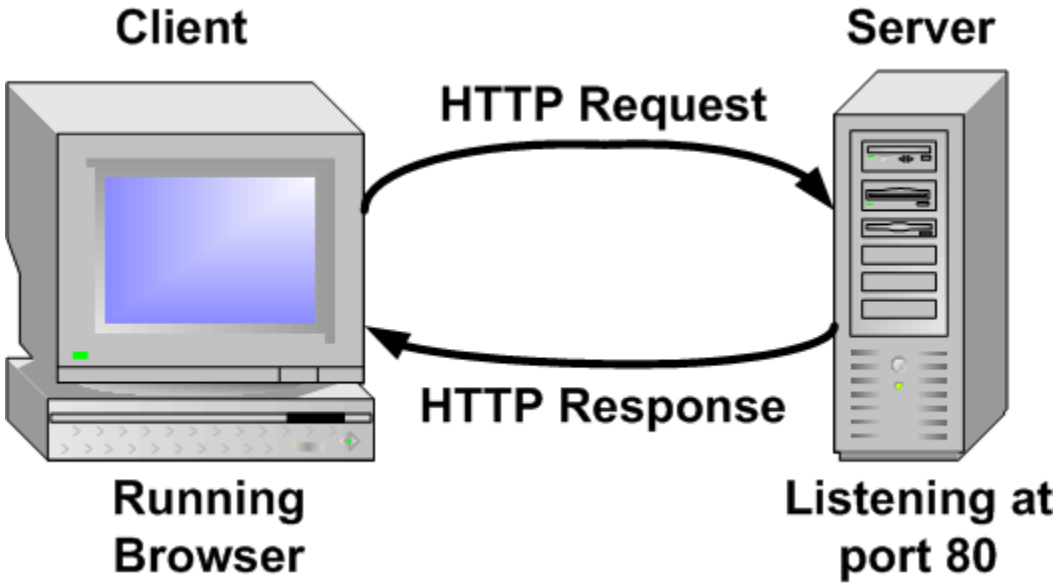
Cilj ovog rada je dati pregled najvećih ranjivosti HTTP protokola, objasniti kako se one mogu zloupotrijebiti, te kako se zaštititi i dati pregled najčešćih HTTP napada, objasniti kako se oni izvode, te kako se od njih zaštititi.
To može biti korisno prosječnim korisnicima u zaštiti vlastitih podataka, WEB sustavima u zaštiti sustava, podataka i mrežnih resursa, te računalnim forenzičarima za forenzičke analize.

2. Kratki pregled HTTP protokola

HTTP (eng. *Hypertext Transfer Protocol*) protokol je aplikacijski protokol OSI modela, koji je osnova je podatkovne komunikacije WEB-a (eng. World Wide Web) [16] i jedan od najkorištenijih protokola na svijetu.

Trenutno su u raširenoj uporabi verzije protokola HTTP/1.1 i HTTP/2.0.

Komunikacija HTTP protokolom funkcionira po principu zahtjev-odgovor u okviru klijent-poslužitelj arhitekture (slika 1).
Najčešći klijenti su internet preglednici, mobilne aplikacije, klijentske aplikacije itd, a najčešći poslužitelji su WEB aplikacije.



Slika 1. HTTP zahtjev-odgovor u kontekstu klijent-poslužitelj arhitekture [20]

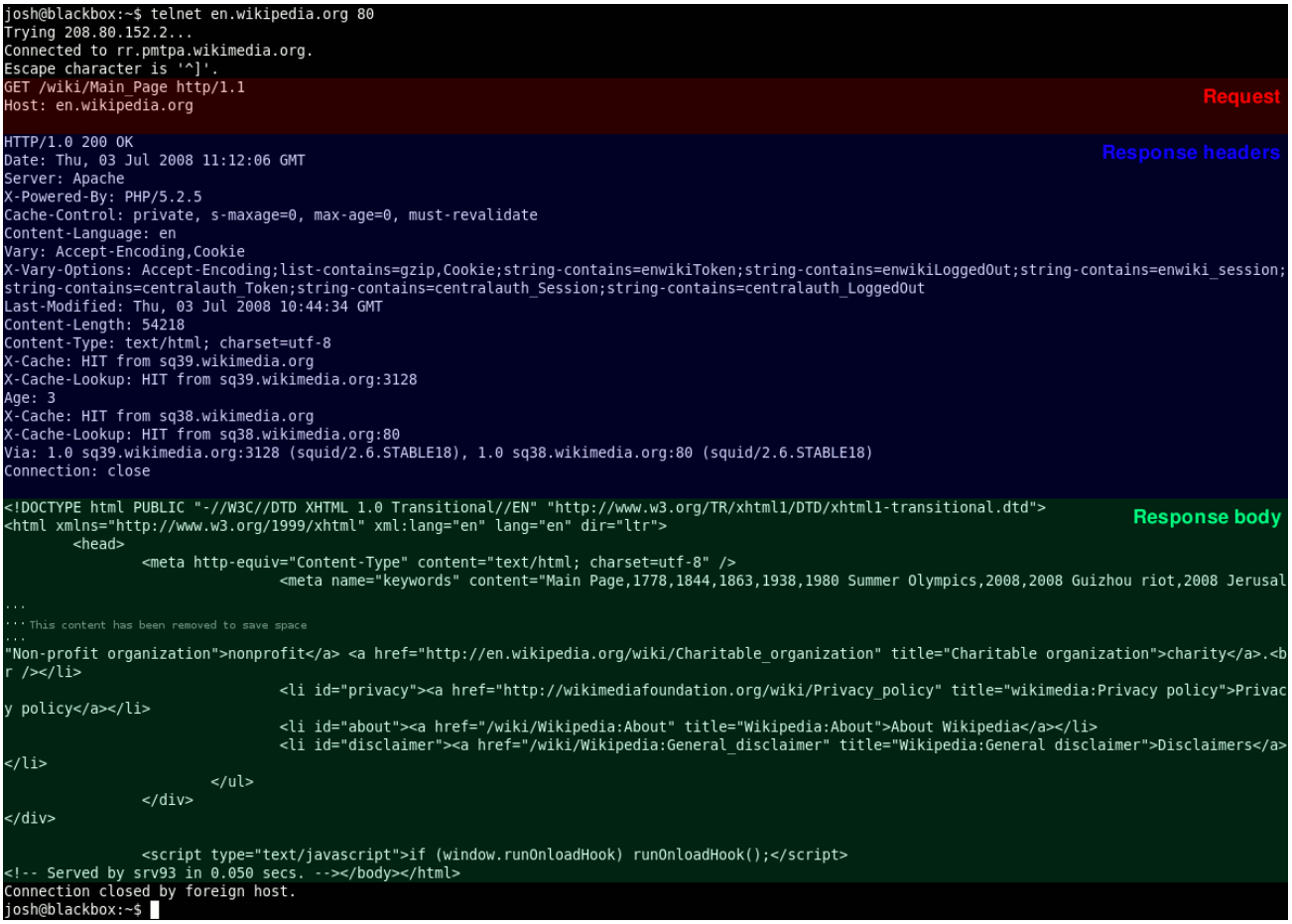
Komunikacija HTTP protokolom se odvija tako da klijent uputi HTTP zahtjev za nekim resursom (npr., HTML stranica ili operacija pretraživanja) i od poslužitelja dobije HTTP odgovor.

Zahtjev sadrži informacije o identifikaciji resursa (odnosno *URL* resursa), verziju protokola (npr., *HTTP/1.1*), HTTP metodu (*GET, POST, PUT, DELETE, HEAD, OPTIONS ili TRACE*), zaglavlja zahtjeva koja definiraju parametre komunikacije [17] (npr., *Accept-Language: en, Server: Apache 1.1*) i tijelo zahtjeva s podacima koji se šalju na poslužitelj (npr., korisnički podaci iz WEB forme u *JSON* formatu).

Odgovor sadrži informacije o verziji protokola, statusni kod (npr., *200 OK*) koji klijentu zahtjeva daje informacije o procesu obrade zahtjeva (100 – informacije dok obrada zahtjeva još traje, 200 – Uspjeh, 300 – Preusmjeravanje, 400 – pogreška klijenta, 500 – pogreška na poslužitelju [18]), zaglavlja odgovora (npr. *Content-Type: text/html*) i tijelo

odgovora s podacima koje poslužitelj vraća klijentu (npr., HTML stranica ili podaci iz baze podataka).

Detaljniji primjer HTTP komunikacije je prikazan na slici 2.



Slika 2. Primjer HTTP komunikacije zahtjev – odgovor [19]

HTTP kao protokol transportnog sloja najčešće koristi TCP protokol, a kao mrežni protokol IP protokol.

3. Ranjivosti HTTP protokola

U okviru HTTP protokola postoje brojne ranjivosti, koje se mogu zloupotrijebiti ili iskoristiti pri forenzičkoj analizi.
Općenito, bilo kakav neovlašteni pristup privatnim, osjetljivim ili zaštićenim informacijama korisnika može se smatrati zloupotrebom tih podataka. Tako se zloupotrebom ranjivosti HTTP protokola može pristupiti takvim informacijama korisnika, poput korisničkih imena, lozinki, osobnih podataka, obrazaca interesa i pretraživanja korisnika, itd.

Zaglavlje i tijelo HTTP poruka pohranjuje podatke u čistom tekstu [2], koji nije kriptiran, čime je svima koji mogu pristupiti HTTP zahtjevu ili HTTP odgovoru omogućen slobodan pristup podacima HTTP poruke.
To nije problem pri ovlaštenom pristupu porukama, odnosno kada porukama pristupaju klijent i namjeravani poslužitelj HTTP poruke.

Problem nastaje u slučaju HTTP napada, kada treća strana ostvari neovlašteni pristup HTTP porukama. Pri tome, budući da su podaci HTTP poruka pohranjeni u čistom tekstu, napadač može ostvariti slobodni, neovlašteni pristup tim podacima, odnosno te podatke zloupotrijebiti.
Tako je moguće pristupiti različitim osjetljivim podacima korisnika: kolačićima za autentifikaciju, korisničkim imenima i lozinkama, ostalim informacijama iz WEB formi [2], osobnim informacijama korisnika, npr., lokaciji , e-mail adresi, enkripcijskim ključevima [1], informacijama s kreditnih kartica, itd.

Postoje i brojne mane HTTP protokola i okružujućeg ekosustava, koje se mogu zloupotrijebiti i bez da se nužno ostvari neovlašteni pristup HTTP prometu preko HTTP napada.
Najčešće se radi o analizi HTTP prometa ili sadržaja zaglavlja HTTP poruka kako bi se otkrile informacije o korisnicima tog HTTP prometa.

Log datoteke
Najveći broj korištenih HTTP poslužitelja, WEB preglednika i proxy poslužitelja zapisuje neke informacije o posluženom HTTP prometu u log datoteke (tablica 1). Najčešće su to [5]: URL zahtjeva, HTTP metoda, verzija HTTP protokola, HTTP statusni kod, IP adresa klijenta, neka HTTP zaglavlja iz zahtjeva (npr., *Referer, User-Agent, Server*) i dr.

Zapisi log datoteke
[10/Dec/2015:16:13:15 +0100] "GET /lib/images/smileys/icon_question.gif HTTP/1.1" 200 5962015-12-10T16:13:15.081735+01:00 rina APACHE2-archiware: 161.53.64.118 - -
[10/Dec/2015:16:13:15 +0100] "GET /lib/images/smileys/icon_exclaim.gif HTTP/1.1" 200 5852015-12-10T16:13:15.095389+01:00 rina APACHE2-archiware: 161.53.64.118 - -
[10/Dec/2015:16:13:15 +0100] "GET /lib/images/smileys/icon_lol.gif HTTP/1.1" 200 7592015-12-10T16:13:15.102425+01:00 rina APACHE2-archiware: 161.53.64.118 - -
[10/Dec/2015:16:13:15 +0100] "GET /lib/images/smileys/fixme.gif HTTP/1.1" 200 8652015-12-10T16:13:15.114552+01:00 rina APACHE2-archiware: 161.53.64.118 - -
[10/Dec/2015:16:13:15 +0100] "GET /lib/images/smileys/delete.gif HTTP/1.1" 200 8632015-12-10T16:13:15.115083+01:00 rina APACHE2-archiware: 161.53.64.118 - -

Tablica 1. Primjer poslužiteljske HTTP log datoteke [22]

Analizom takvih zapisa iz log datoteka mogu se otkriti različite informacije o korisnicima radi zloupotrebe ili forenzičke analize.

TKO?

Jedno od temeljnih forenzičkih pitanja jest TKO je nešto napravio. Tako je često polazna točka u analizi HTTP prometa odrediti tko je uputio HTTP zahtjev, tj., povezati dostupni HTTP promet s korisnicima. Na taj se način svaka dodatna analiza HTTP prometa, odnosno otkrivene informacije mogu povezati s konkretnim korisnikom.
Jedna od najjednostavnijih metoda otkrivanja korisnika zahtjeva je analiza IP adresa iz log datoteka. Budući da se IP adrese korisnika relativno rijetko mijenjaju, IP adresa se može povezati s konkretnim korisnikom.

Nakon što je poznat korisnik koji je uputio HTTP zahtjeve, moguće je dodatnom forenzičkom analizom HTTP log datoteka otkriti brojne informacije o korisniku. Potrebno je naglasiti kako su ovo privatne i osobne informacije korisnika, te se ne bi trebale koristiti bez njihova dopuštenja.

Analiza URLova

Tako se analizom URL-ova istog korisnika, tj., stranica koje je korisnik posjećivao, (npr., iz log datoteka WEB preglednika) mogu otkriti navike pretraživanja (eng. *reading pattern*) i područja interesa korisnika [1].

Analizom URL-ova se mogu saznati i osjetljive korisničke informacije (npr., lozinke), ako se za slanje podataka na poslužitelj koristi HTTP GET zahtjev, budući da se podaci poslani HTTP GET zahtjevom zapisuju i šalju u URL-u zahtjeva, a URL se često zapisuje u log datoteke.

Stoga se preporučuje slanje podataka na poslužitelj isključivo HTTP POST i HTTP PUT metodama, kod kojih se poslani podaci šalju u tijelu HTTP zahtjeva. Također, preporučuje se da poslužitelju uopće ne podržavaju prihvat podataka HTTP GET zahtjevima, posebno ako se radi o WEB formama.

Analiza HTTP zaglavlja

Analizom nekih HTTP zaglavlja mogu se također saznati informacije o korisniku [1].

Zaglavlja *Server*, *Via* i *User-Agent* sadrže informacije o poslužitelju, proxy-ju ili pristupnoj klijentskoj aplikaciji (npr., *Mozilla/<version>* za WEB preglednik [6]) korisnika [4].
Te se informacije mogu dalje iskoristiti, npr., informacije o vrsti poslužitelja ili proxy-ja (npr. *Apache/1.1*) se mogu zloupotrijebiti za različite napade, ako taj poslužitelj ima poznate ranjivosti.

Zaglavlje *Referer* sadrži informaciju (URL) o prethodnoj stranici koju je korisnik posjetio, tj., o stranici s koje je putem hiperveze (eng. *link*), došao na trenutnu stranicu [3].
Analizom *Referer* zaglavlja HTTP zahtjeva istog korisnika pristiglih na neki poslužitelj (npr. iz HTTP log datoteke poslužitelja) se mogu otkriti stranice koje je korisnik posjećivao, čime se mogu saznati navike pretraživanja i područja interesa korisnika.

Dodatne se informacije o korisniku mogu saznati iz zaglavlja *From* i *Accept-language.*
Zaglavlje *From* sadrži e-mail adresu korisnika odgovornog za automatiziranu operaciju koja je generirala HTTP promet koji se analizira.
Zaglavlje *Accept-language* sadrži podatke o jezicima koje korisnik razumije, iz čega se mogu zaključiti osjetljive i privatne informacije o nacionalnoj i etničkoj pripadnosti korisnika [1].

Mehanizmi prevencije zloupotrebe ovih informacija često uključuju: informiranje korisnika o podacima koji se šalju, dopuštanje korisniku da odabere hoće li se slati podaci (npr., za jezik korisnika), dopuštanje korisniku da modificira podatke koji se šalju itd.

4. Napadi

Postoje različite vrste napada, kojima se mogu iskoristiti ranjivosti HTTP protokola.

4.1. Napad na datotečni sustav

Neki HTTP poslužitelji poslužuju datoteke iz datotečnog sustava na HTTP zahtjeve [1]. Pri tome se često URL zahtjeva na neki način preslikava u putanju tražene datoteke na disku (npr., http://www.host.com/images/5 -> C:/{putanja do radnog direktorija}/images/image5.jpg).

Pri tome je namjera poslužitelja posluživati isključivo datoteke koje se nalaze na jednom dijelu datotečnog sustava (npr., isključivo datoteke iz radnog direktorija poslužitelja), dok ostatak datotečnog sustava treba ostati nedohvatljiv.
Ako poslužitelj nije dovoljno dobro sigurnosno podešen, ovakvu je konfiguraciju moguće zloupotrijebiti napadom. Pri tome je cilj napadaču dohvatiti datoteke s dijela datotečnog sustava koji bi trebao ostati nedohvatljiv (npr., izvan radnog direktorija poslužitelja).

Općenito je cilj napadača konstruirati HTTP zahtjev s takvim URL-om kojega će poslužitelj preslikati u putanju datoteke koja bi trebala biti nedohvatljiva, te dohvatiti takvu datoteku, uobičajeno HTTP GET zahtjevom.
Tako napadač može dohvatiti privatne datoteke korisnika izvan radnog direktorija poslužitelja, log datoteke poslužitelja, konfiguracijske datoteke, izvorni kod poslužitelja i dr.

Česta taktika pri ovoj vrsti napada je korištenje modifikatora za relativne putanje datotečnog sustava, poput modifikatora „..“ kojim se na UNIX, Windows i drugim operacijskim sustavima pristupa naddirektoriju trenutnog direktorija. Primjer dopuštenog (zeleno – pristup slici u radnom direktoriju) i nedopuštenog (crveno – pristup tekstualnoj datoteci s lozinkama izvan radnog direktorija) pristupa je dan u tablici 2.

HTTP metoda	**URL**	**Relativna putanja na datotečnom sustavu (u odnosu na radni direktorij poslužitelja)**	**Apsolutna putanja na datotečnom sustavu**
GET	/images/image5.jpg	/images/image5.jpg	C:/{putanja do radnog direktorija}/images/image5.jpg
GET	/images/../../lozinke.txt	/../lozinke.txt	C:/{putanja do naddirektorija radnog direktorija}/lozinke.txt

Tablica 2. Primjer HTTP napada na datotečni sustav

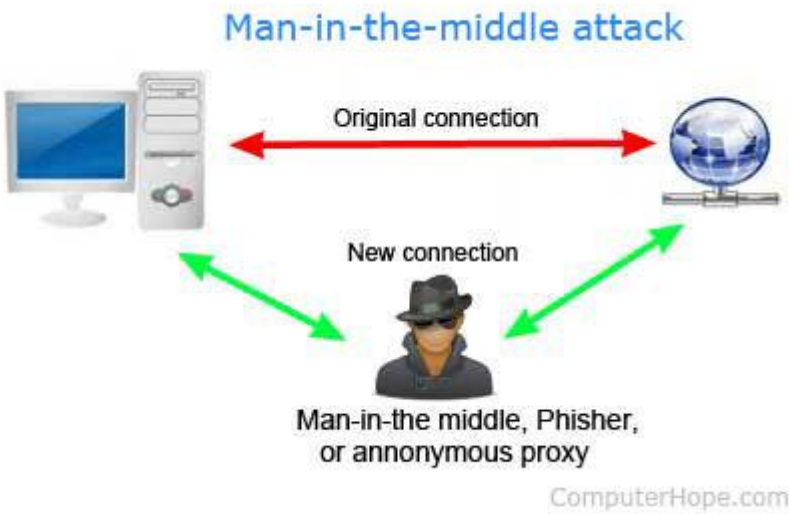
Sustav se može zaštititi od ovakvih napada tako da ograniči format URL-a koji se prihvaća na poslužitelju, npr., tako da ne dopušta modifikatore relativnih putanji (npr. „..“) unutar URL-a. Dodatno je posebno potrebno zaštititi poslužiteljske datoteke (konfiguracijske datoteke, izvorni kod, datoteke s lozinkama,…) od vanjskog pristupa.

Ipak, unatoč mehanizmima zaštite, iskustvo pokazuje da i najmanje pogreške u implementaciji ovakvih HTTP poslužitelja mogu dovesti do ranjivosti u sigurnosti sustava [1].

4.2. Posrednički napadi

HTTP protokol je protokol koji **ne pamti stanje** (eng. *stateless*), čime je posebno pogodan za **posredničke** (eng. *man in the middle*) napade [2], budući da napadač u ulozi posrednika ne mora ostvariti trajnu vezu (eng. *connection*).

Općenito se posrednički napad ostvaruje tako da [8] treća strana neovlašteno prima i prosljeđuje HTTP zahtjeve između klijenta i poslužitelja, koji misle da direktno komuniciraju jedan s drugim i ne znaju da napadač uopće postoji (slika 3).



Slika 3. Posrednički napad [7]

<u>TCP posrednički napad</u>

Najčešći oblik ostvarivanja uspješnog HTTP posredničkog napada jest napad na TCP veze koje koriste HTTP zahtjevi i odgovori. Nakon što različitim tehnikama ostvari ovlasti nad TCP vezom između klijenta i poslužitelja, napadač podijeli originalnu TCP vezu na dvije nove veze (klijent-napadač, napadač-poslužitelj) [21], kao što je prikazano na slici 3.
Od tog trenutka, napadač može neopaženo prisluškivati HTTP promet između klijenta i poslužitelja. Svi HTTP zahtjevi upućeni poslužitelju (od klijenta) i svi HTTP odgovori upućeni klijentu (od poslužitelja) će biti poslani napadaču, koji im može pročitati sadržaj. Ako napadač želi ostati neopažen, dovoljno je da primljene HTTP zahtjeve transparentno proslijedi poslužitelju, a HTTP odgovore klijentu.

Proxy napadi

Još jedna vrsta posredničkih napada su proxy napadi, tj., napadi koji iskorištavaju pristup proxy poslužiteljima [1].

Proxy poslužitelji su prirodni posrednici HTTP prometa, koji ovlašteno prosljeđuju promet između klijenta i poslužitelja prometa, najčešće radi nadzora i filtriranja, povećanja performansi, prevođenja, anonimnog pristupa, sigurnosti, strukturiranja raspodijeljenih sustava i dr. [9].

Pri tome, proxy poslužitelji često zapisuju velike količine podataka o HTTP prometu u log datoteke.
Problem nastaje kada napadač ostvari neovlašteni pristup proxy poslužitelju, odnosno podacima kojima se može pristupiti sa proxy poslužitelja.
U tom slučaju, napadač može pristupiti svim podacima iz HTTP poruka, budući da proxy poslužitelj prosljeđuje poruke koje su u čistom tekstu, tj., nisu kriptirane, analizirati zahtjeve iz log datoteka proxy poslužitelja ili, kod keširajućih proxy poslužitelja pristupiti keširanom sadržaju, koji često sadrži osobne i osjetljive korisničke informacije [1].

4.3. HTTP DoS napadi

DoS (eng. *Denial of service*) napadi su napadi kod kojih napadač želi privremeno ili trajno poremetiti usluge koje pruža neki poslužitelj, kako bi onemogućio pristup tim uslugama od strane redovnih korisnika tih usluga [10].

Napadi poplavom (eng. *Flood*)

Najčešće korištena vrsta DoS napada su napadi poplavom (eng. *flood*).
Kod takvih napada napadač pokušava preplaviti HTTP poslužitelj zahtjevima ili odgovorima, tako da on bude nedostupan za legitimne zahtjeve korisnika.
Zbog toga poslužitelji često koriste zaštitne uređaje ili usluge koji odbacuju neispravni HTTP promet prije nego što dospije do poslužitelja, pa napadači moraju i to uzeti u obzir.

Više je poznatih HTTP napada poplavom [11].

Poplava smećem (eng. Garbage flood)

Napadač šalje neispravne binarne podatke na HTTP port (npr., 80 ili 443), kako bi preplavio interne međuspremnike WEB poslužitelja i zaštitnih uređaja. Tako ti međuspremnici postaju nedostupni za legitimni HTTP promet.
Kako se zaštitni uređaji većinom fokusiraju na zaštitu od valjanog HTTP prometa, ova vrsta napada se često previdi u zaštiti.

Poplava valjanim HTTP zahtjevima

Napadač šalje veliku količinu valjanih HTTP (*GET, POST, PUT, DELETE, HEAD i OPTIONS*) zahtjeva na poslužitelj, za sve usluge koje nudi poslužitelj. Zahtjevi su valjani i u skladu

sa svim normama, tako da prolaze zaštitne uređaje. Cilj napada je preplaviti usluge poslužitelja lažnim zahtjevima, kako bi one postale spore ili nedostupne za legitimne zahtjeve redovnih korisnika.

Nedostatak opisanih napada je taj što zahtjevaju veliku količinu generiranog napadačkog prometa. To je skupo, budući da se najčešće ostvaruje raspodijeljenim DoS napadom, sa više napadačkih poslužitelja.

S druge strane, postoje HTTP napadi poplavom koji ne zahtjevaju veliku količinu HTTP prometa, pa su jeftiniji i jednostavnije ih je izvesti.

Poplava nevaljanim konstruktima unutar HTTP zahtjeva

Napadač šalje nevaljane konstrukte u inače valjanim HTTP zahtjevima, pokušavajući tako srušiti poslužitelj koji ne očekuje i ne zna obraditi takve nevaljane konstrukte. Nevaljani konstrukti mogu biti različiti. Primjeri uključuju: slanje G3T umjesto GET zahtjeva, slanje HTTP/1,1 umjesto HTTP/1.1 za verziju protokola itd.

Poplava sporim HTTP zahtjevima (eng. low and slow attack)

Napadač šalje valjani HTTP GET zahtjev na poslužitelj, ali ga šalje jako sporo, odnosno u jako puno paketa protokola nižeg sloja (TCP/IP).
Napadač iskorištava svojstvo da je obično broj istovremenih TCP veza dostupnih poslužitelju ograničen, pa mu je cilj držati vezu sa poslužiteljem otvorenom što je duže moguće, te tako onemogućiti korištenje te veze od strane legitimnih korisnika.
Npr., napadač može sporo slati verziju HTTP protokola. Tako za verziju protokola HTTP/1.1 u jednom paketu šalje H, u drugom T, u trećem T, u četvrtom P, u petom /, u šestom 1, itd. Budući da specifikacija HTTP protokola dopušta ovakvo ponašanje, poslužitelj mora držati vezu otvorenom dok ne posluži čitav zahtjev. Kod većih zahtjeva to može trajati jako dugo i za to vrijeme redovni korisnici ne mogu koristiti tu vezu za legitimni HTTP promet.

4.4. Napadi ubacivanja koda (eng. *Injection*)

Postoje različite vrste napada koji se izvode tako da napadač na neki način ubaci maliciozni kod u sustav. Maliciozni kod se potom izvodi na sustavu, te na neki način čini štetu sustavu (npr., dohvati zaštićene podatke, sruši sustav, obriše bazu podataka i dr.).

HTTP protokol dopušta širok format URL-a i podataka u tijelu HTTP zahtjeva, čime je omogućeno slanje malicioznog koda unutar parametara iz WEB formi [13] poslanih u URLu ili u tijelu zahtjeva.

Sljedeći primjer [12] u tablici 3 ilustrira jednostavan *SQL injection* napad.
Ovaj napad iskorištava ranjivost HTTP protokola da ne nudi podršku za tipove URL

parametara i validaciju tipova parametara.
Drugim riječima, kao vrijednost parametra *id* se može predati namjeravani tip podataka (tablica 3 – redak 1 – id=532) kojega poslužitelj očekuje, ili nenamjeravani tip podataka (tablica 3 – redak 2 – id=*niz znakova*), koji može sadržavati maliciozni kod.

Kod na poslužitelju	**HTTP parametar**	**HTTP URL**	**Rezultirajući SQL kod**	**Rezultat izvršavanja**
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";	id	http://example.com/app/accountView?id=532	SELECT * FROM accounts WHERE custID='532';	Uspješan i legitiman dohvat zapisa o računima od korisnika s id=532
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";	id	http://example.com/app/accountView?id=' or '1'='1	SELECT * FROM accounts WHERE custID='' or '1'='1';	Uspješan *SQL injection* napad. Nelegitiman dohvat svih zapisa iz relacije *accounts*.

Tablica 3. Primjer *SQL injection* napada

Ozbiljniji *SQL injection* napadi od primjera iz tablice 3 mogu dovesti do ozbiljnijih posljedica na sustavu, poput brisanja podataka, izvođenja pohranjenih procedura, rušenja sustava, itd. [12].

Zaštita od ovakvog napada se ne može riješiti izmjenom HTTP protokola, nego je odgovornost validacije primljenih podataka iz WEB formi na aplikacijama koje primaju i obrađuju te podatke.

Najčešći primjeri ovog napada su [12]: SQL, NoSQL, OS naredbe, *Object Relational Mapping* (ORM), LDAP, *Expression Language* (EL) i *Object Graph Navigation Library* (OGNL) ubrizgavanje koda.

4.5. HTTP preopterećivanje spremnika (eng. *buffer overflow)*

Na sličan način se preko HTTP zahtjeva može izvesti *buffer overflow* napad na sustav.
Kod ovog napada se preko WEB formi ili URL-a HTTP zahtjeva [15] na poslužitelj šalju podaci tako da se za neki parametar pošalje veća količina podataka, nego što poslužitelj može primiti za taj parametar (npr., za polje za koje je na poslužitelju alocirano 30 bajtova se pošalje 31 bajt podataka [14]).
Tako ova vrsta napada iskorištava svojstvo (ranjivost) HTTP protokola da ne nudi podršku za ograničavanje dužine vrijednosti poslanih podataka.
Cilj napada je izazvati pogrešku (*error*, *exception*, *buffer overflow*) na sustavu pri obradi pristiglih podataka i srušiti sustav.
Obranu od ovakve vrste napada ne može pružiti HTTP protokol, nego je odgovornost obrane na validaciji pristiglih podataka na poslužitelju.

5. Zaključak

HTTP protokol je jedan od najkorištenijih telekomunikacijskih protokola današnjice, budući da je HTTP protokol osnova za svu komunikaciju na WEB-u.

Budući da je HTTP tako rasprostranjen protokol, konstantno se radi na otkrivanju propusta u njemu i na njegovom poboljšanju. Ipak, postoje brojne ranjivosti u okviru HTTP protokola, koje se mogu zloupotrijebiti ili poslužiti za forenzičku analizu.

HTTP pohranjuje podatke u čistom tekstu, pa svatko tko ima pristup HTTP prometu može pročitati sadržaj HTTP poruka, koji često može sadržavati osjetljive podatke, poput lozinki i osobnih podataka.

HTTP klijenti i poslužitelji zapisuju velike količine podataka o HTTP prometu u log datoteke, koje se zajedno s nekim zaglavljima koja HTTP protokol koristi (*Server, Via, From, User-Agent, Referer, From, Accept-language* i dr.) mogu iskoristiti u analizi kako bi se otkrile brojne privatne i osjetljive korisničke informacije, poput navika pretraživanja i područja interesa te etničke pripadnosti korisnika.

HTTP protokol je također ranjiv na različite vrste napada, poput posredničkih napada, HTTP DoS napada, te različitih napada ubacivanja malicioznog koda.

Zaštita od iskorištavanja ranjivosti se ne može uvijek postići poboljšanjem HTTP protokola, nego je često odgovornost implementacije zaštite na poslužitelju. Pri tom je potrebno poznavati ranjivosti HTTP protokola i o njima voditi računa pri implementaciji zaštite sustava, npr., korištenjem preporučenih dobrih praksi, validacijom podataka, te informiranjem korisnika sustava.

LITERATURA

[1] https://www.w3.org/Protocols/rfc2616/rfc2616-sec15.html - HTTP/1.1: Security Considerations
[2] http://k12linux.mesd.k12.or.us/cascadelink/text7.htm - Weaknesses of HTTP
[3] https://en.wikipedia.org/wiki/HTTP_referer
[4] https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html
[5] https://httpd.apache.org/docs/1.3/logs.html
[6]https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent
[7] https://www.computerhope.com/jargon/m/maninthemiddleattack.jpg
[8] https://en.wikipedia.org/wiki/Man-in-the-middle_attack
[9] https://en.wikipedia.org/wiki/Proxy_server
[10] https://en.wikipedia.org/wiki/Denial-of-service_attack
[11] https://blog.radware.com/security/2017/11/http-attacks/
[12] OWASP Top 10 - 2017 The Ten Most Critical Web Application Security Risks - https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf
[13] http://searchsecurity.techtarget.com/tip/HTTP-attacks-Strategies-for-prevention
[14] https://www.owasp.org/index.php/Buffer_Overflow
[15] http://searchsecurity.techtarget.com/tip/HTTP-attacks-Strategies-for-prevention
[16] https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[17] https://en.wikipedia.org/wiki/List_of_HTTP_header_fields
[18] https://hr.wikipedia.org/wiki/HTTP
[19]https://upload.wikimedia.org/wikipedia/commons/c/c6/Http_request_telnet_ubuntu.png
[20]https://easyphpprogramming.files.wordpress.com/2014/04/client_server_interaction.gif
[21] Računalna forenzika, predavanja, Mrežna Forenzika, Kristian Skračić, Predrag Pale