CARNet
HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Honeypot – zamke na Internetu

NCERT-PUBDOC-2012-07-337

Nacionalni CERT

Sadržaj

1 Uvod

U svijetu računalne sigurnosti termin honeypot označava zamku postavljenu napadačima. Cilj te zamke je uhvatiti napadača i analizirati njegove tehnike napada. Honeypot se obično sastoji od jednog ili više računala koja su podešena tako da se napadačima predstavljaju kao sustavi na kojima postoje vrijedne informacije i potiču ih na napad. U stvarnosti su ta računala strogo izolirana i kontrolirana te omogućuju praćenje svih napadača koji ih pokušaju kompromitirati.

Honeypote često postavljaju antivirusne kompanije i specijalizirane udruge. No, honeypot može postaviti gotovo svatko tko ima pristup Internetu. Korist od njih je višestruka. Honeypoti omogućuju analizu napadačkih tehnika kriminalaca. Time sigurnosni stručnjaci dobivaju informaciju o tome što je aktualno i koje su najnovije prijetnje na Internetu. U prvom redu to se najviše odnosi na prikupljanje novih ranjivosti i novih primjeraka zlonamjernog koda.

Različite vrste honeypota se dijele u skupine s obzirom na to koji stupanj interakcije dopuštaju napadaču. Prema stupnju interakcije postoje:

- Honeypoti visokog stupnja interakcije – dopuštaju napadaču potpuni pristup računalu
- Honeypoti niskog stupnja interakcije – ne dopuštaju potpuni pristup računalu već samo emuliraju servise i njihove ranjivosti

Honeypote također možemo podijeliti prema tome koriste li se oni kao poslužitelji ili kao klijenti. Honeypoti koji su dizajnirani kao poslužitelji služe prikupljanju zlonamjernih programa i napada koji se automatski šire mrežom. Honeypoti dizajnirani kao klijenti prvenstveno služe prikupljanju zlonamjernih programa s web stranica. Oni simuliraju surfanje webom i analiziraju svaku posjećenu web stranicu.

Ovaj dokument donosi opis nekoliko implementacija i poslužiteljskih i klijentskih honeypota. Za svaki honeypot opisan je njegov način rada i mogućnosti koje on pruža.

2 Poslužiteljski honeypoti

Poslužiteljski honeypoti su ona vrsta honeypota koja se instalira na neki poslužitelj i onda čeka napadače. Oni emuliraju popularne mrežne servise i ranjivosti u njima u nadi da će ih napadači pokušati iskoristiti. Ovakvi honeypoti su dobri u skupljanju malvera koji se automatski širi po internetu.

Skupljanje i analiza malvera važan je dio računalne sigurnosti. Neki od razloga zašto sigurnosne kompanije i stručnjaci analiziraju malver su:

- Proučavanjem malvera dobivamo više informacija o tome kako se zaštiti protiv određene ranjivosti. Sustavi za otkrivanje uljeza i antivirusni programi mogu osvježiti listu potpisa koji se koriste za otkrivanje malicioznih datoteka i mrežnog prometa.
- Možemo iskoristit tehniku skupljanja malvera kako bi preventivno zaštitili mrežu. Dakle, sustav za prikupljanje malvera može uz sustav za otkrivanje uljeza štiti mrežu u stvarnom vremenu.
- Skupljanje malvera u velikoj količini (npr. *honeynets*) nam može poslužiti kako bi generirali statistike o načinima napada, trendovima napada i slično

No, skupljanje malvera je zahtjevan posao, pogotovo kada se uzme u obzir rastući broj malvera na Internetu i brzina kojom se oni šire. Honeypoti nude visoki supanj automatizacije ovog procesa kako bi istraživačima olakšali posao i omogućili im da prikupe što više primjeraka malvera.

2.1.1 Nepenthes

Nepenthes je bio popularan honeypot namjenjen skupljanu malvera. Cijeli kod je dostupan na URL-u: http://nepenthes.carnivore.it/. Danas je nephentes zastario i sami autori preporučuju korištenje Dionaea honeypota (opisan kasnije u dokumentu). No, u nekim situacijama nepenthes može biti koristan.

Arhitektura Nepenthes honeypota je ostvarena kroz modularan dizajn i ovakav dizajn je postao *de facto* standard za izgradnju honeypota. Moduli su entiteti koji obavljaju posao skupljanja, pohrane i obrade malvera. Jezgra nepenthesa upravlja mrežnim sučeljima (I/O operacijama) i koordinira akcije modula. Nepenthes se sastoji od nekoliko tipova modula:

Moduli emulatori servisa – emuliraju ranjive dijelove mrežnih servisa. Ovo je ključ efikasnosti nepenthesa koji umjesto emuliranja cijelog servisa ili sustava, emulira samo potrebne dijelove. Ovaj modul pokušava prevariti napadače glumeći ranjivosti.

Moduli za parsiranje shell koda - analiziraju podatke primljene od strane jednog od modula emulatora servisa. Ovi moduli parsiraju primljeni *shell* kod ili strojni kod, te izvlače informaciju o propagirajućem malveru.

Moduli dobavljači – koriste informaciju o malveru koju su primili od modula za parsiranje koda kako bi dobavili malver s udaljenog računala. Primljeni URLovi ne trebaju nužno biti HTTP ili FTP URLovi, nego se može raditi o TFTP ili drugom protokolu.

Moduli pohrane – mogu spremiti malver kao binarnu datoteku na disku, u bazu podataka ili poslati malver antivirusnim kompanijama.

Moduli logiranja – logiraju informacije o emuliranim procesima i o dobavljenom malveru. Pomažu u dobivanju pregleda o tome kako je napad tekao na našem honeypotu.

Modularna arhitektura je lako i brzo prilagodljiva promjenama. Ako želimo ugraditi novi modul ili zamijeniti stari na bilo kojem od 5 navedenih nivoa modula, jedino je potrebno prilagoditi novi modul sučeljima.

2.1.2 Honeytrap

Honeytrap je skupljač malvera koji procese dohvata podataka i analize napada radi odvojeno. Dostupan je s URL-a: http://sourceforge.net/projects/honeytrap/

Proces dohvata podatka se u potpunosti odvija u jezgri honeytrapa. Daljnja analiza napada, kao što je provjera tijeka napada se obrađuje sa dodacima (*plug-in*) koji se mogu učitavati tijekom rada honeypota. Nije potrebno gasiti ili ponovno prevoditi Honeytrap softver kao u slučaju Nepenthesa.

Honeytrap nema klasični pristup skupljanju malvera gdje se emuliraju dobro poznate ranjivosti u servisima. Ovakav pristup ne radi ako želimo promatrati nove i nepoznate napade (*zero-day attacks*). Jezgra Honeytrap sustava detektira zahtjev za novom konekcijom na slobodni TCP port, pokreće poslužiteljski proces koji otvara ovaj TCP port i upravlja dolazećim podacima. Na ovaj način možemo prihvatiti konekcije za bilo koji napad bez obzira je li on do sada bio nepoznat. Zbog dinamičkog otvaranja portova nije potrebno držati tisuće pokrenutih procesa i otvorenih portova u isto vrijeme kako bi hvatali nove napade. Dinamičko uspostavljenje sjednica se može riješiti na dva načina:

- *Emulacija servisa*: Koncept emulacije servisa je nešto jednostavniji nego kod Nepenthesa. Honeytrap može čitati iz datoteke odgovor na zahtjev koji dolazi na određeni port. Datoteke sadrže odgovore snimljene promatrajući sjednicu stvarnog servisa, te su sve razmjene poruka spremljene u odgovarajući direktorij. Dodavanje nove emulacije servisa znači da je potrebno snimiti promet stvarnog servisa i spremiti u datoteku u odgovarajućem formatu.
- *Zrcalni način rada*: Ovaj način dinamičkog uspostavljanja sjednice je najzanimljiviji dodatak Honeytrap softvera. Svi dolazni podaci se šalju nazad napadaču, tako da se honeypot ponaša kao zrcalo. To znači da honeypot pokušava uspostaviti TCP sjednicu sa zlonamjernim računalom na istom TCP portu koji je odredišni port u dolazećoj sjednici. Odgovor od zrcalne konekcije sa šalje nazad na inicijalnu konekciju i obrnuto. Ovakav način rada u efektu izgleda kao da zlonamjerno računalo napada samo sebe. Ovo radi u praksi, jer ako je neko računalo zaraženo nekim malverom koji pokušava propagirati po internetu, tada je i sporno računalo ranjivo.
- *Posrednički način rada*: U ovom načinu rada svi dolazeći podaci se prosljeđuju drugom računalu ili servisu, tako da se Honeytrap ponaša kao posrednički poslužitelj. Dodatno, Honeytrap sav promet sprema za kasniju analizu.
- *Ignoriranje dolazećih paketa*: Svi dolazeći paketi se ignoriraju. Ova mogućnost se koristi kako bi blokirali pristup na određene portove ako ih ne želimo procesirati kroz Honeytrap.

Drugačiji načini rada mogu biti konfigurirani posebno za svaki port. Sjednice kojima trebaju upravljati drugi honeypoti ili stvarni servisi mogu biti proslijeđene k njima, dok druge mogu biti upravljane kroz emulaciju servisa ili zrcalni način rada.

Svi podaci procesirani kroz Honeytrap se spremaju na datotečnom sustavu. Oni mogu biti obrađeni kroz nekoliko različitih dodataka za automatsku analizu. Npr. mogu se parsirati u potrazi za *shell* komandama koje dohvaćaju malvera. Dodatno, dodaci prepoznaju FTP i TFTP komande i automatski dohvaćaju resurse. Sljedeći dodaci su dostupni:

- Osnovni modul (jezgra Honeytrapa) koja sprema kompletni napad u datotečni sustav za naknadnu analizu sa dodatnim alatima.
- Parser za FTP komande i implementacija FTP klijenta koja nastoji biti slična MS Windows implementaciji jer se najviše iskorištava u napadima.
- Parser za TFTP komande i implementacija TFTP klijenta.
- Parser za HTTP URLove u napadu na VNC servere. Ovdje se datoteke skidaju pomoću `curla` ili `wgeta`.
- Dodatak koji prepoznaje i dekodira *base64* kodiran promet kako bi se pripremio za daljnju analizu.

2.2 Dionaea

Do sad su opisana dva honeypota čije funkcionalnosti su osnova za skupljanje malvera. Dionaea je honeypot koji spaja funkcionalnosti Nepenthesa i Honeytrapa. Riječ je o nešto novijem honeypot-u, razvijenom kao naslijednik nepenthesa. Honeypot je dostupan s URL adrese: http://dionaea.carnivore.it/

2.2.1 Emulacija servisa

- **SMB** – *Server Message Block* je komunikacijski protokol koji se koristi na Microsoft Windows operacijskim sustavima za dijeljeni pristup datotekama, printerima i serijskim portovima. SMB ima veliku povijest ranjivosti koje se mogu iskoristiti s udaljenog računala, te je jako popularna meta među crvima. Osim emulacije ranjivosti Dionaea pruža i mogućnost podizanja (*uploada*) datoteka na dijeljene SMB repozitorije.
- **HTTP** – podržava HTTP protokol na portu 80, te protokol HTTPS, iako ne postoji kod koji obrađuje skupljene podatke. Za HTTPS protokol, stvara se certifikat kojeg honeypot sam potpisuje.
- **FTP** – nudi osnovni FTP poslužitelj na portu 21. Može stvoriti direktorije te skidati i postavljati datoteke.
- **TFTP** – TFTP poslužitelj koji može posluživati datoteke na portu 69.
- **MSSQL** – Ovaj modul implementira Tabular Data Stream protokol koji koristi Microsoft SQL Server. Sluša na TCP portu 1433 i dopušta klijentima autorizaciju.
- **MySQL** – Implementira protokol kojim komunicira MySQL poslužitelj.
- **SIP (VoIP)** – Ovo je VoIP modul honeypota. SIP je VoIP protokol koji se ovdje koristi jer se danas najčešće primjenjuje. Modul može stvoriti novu instancu SIP sjednice i otvoriti potrebne RTP kanale. Modul čeka dolazeću SIP poruku (npr. INVITE), logira sve podatke, čak i RTP promet.

2.2.2 Parsiranje *shell* koda

Parsiranje (tj. otkrivanje) *shell* koda iz mrežnog prometa koji dolazi od strane napadača Dionaea izvodi pomoću `libemu` biblioteke. Ovaj modul može detektirati *shell* kod, odrediti njegovu funkciju te ga izvršiti. Otkrivanje *shell* koda se izvodi pomoću takozvane GetPC heuristike.

Profiliranje (određivanje funkcije) *shell* koda se odvija tako da se spreme svi API pozivi i pripadajući argumenti koji se izvedu u `libemovom` VM-u. Za većinu *shell* koda profiliranje je dovoljno, tj. otkriveni API pozivi i argumenti su dovoljni da bi se otkrilo koje su namjere napadača i na koji način djelovati protiv njih. Za višefazne *shell* kodove, gdje bi prvi *shell* kod povukao drugi *shell* kod, profiliranje nije dovoljno. U tom slučaju je potrebno izvršiti *shell* kod kako bi dokučili sljedeći.

2.2.3 Dobavljanje malvera

Kada je honeypot dobio lokaciju datoteke koju napadač pokušava preuzeti na naše računalo, Dionaea će pokušati dobaviti tu datoteku. Implementirani su FTP i TFTP protokoli za dobavljanje datoteka u pythonu (ftp.py i tftp.py). Dobavljanje datoteka pomoću HTTP protokola je implementirano u `curl` modulu (koristi *libcurl*). Programsko sučelje *libcurl*

također može dobavljati datoteke pomoću FTP protokola, međutim ugrađeni FTP klijent je specijalno dizajniran da oponaša MS Windows FTP klijent.

2.2.4 Pohrana malvera

Kada imao kopiju malvera, želimo ga spremiti lokalno za daljnju analizu ili ga spremiti na udaljeno računalo. Dionaea ima mogućnost prenošenja datoteke pomoću HTTP POST metode na nekoliko servisa kao što su CWSandbox, Norman Sandbox ili VirusTotal.

2.2.5 Logiranje napada

Osim kopije malvera ponekad je poželjno imati i informacije o tome kako je tekao napad. Dionaea može zapisati ovakve informacije u tekstualnu datoteku, iako se ovakav zapis zbog svoje složenosti koristi uglavnom za ispravljanje grešaka u samom softveru. Dionaea podržava interni komunikacijski mehanizam koji se zove incidenti (*incidents*). Koristeći mehanizam incidenta i *ihandlera*, možemo na jednostavan način dobiti informaciju koju tražimo i zapisati ju u format koji sami odaberemo. Tako na primjer *logsql* python skripta zapisuje incidente u sqlite bazu podataka.

2.2.6 Zrcalni način rada – modul nfq

Nfq modul omogućuje zrcalni način rada dionaea honeypota. Nfq modul radi interakciju s jezgrom operacijskog sustava, a nfq python skipta se brine o koracima koji su potrebni da bi se startao novi servis na određenom portu.

O ideji zrcalnog načina rada skupljača malvera je više rečeno u poglavlju o Honeytrapu. Sada će biti spomenuti neki problemi ovog pristupa „emulacije servisa“ koje Dionaea uspješno rješava:

- *port scanning:* Ako netko skenira portove našeg honeypota, on će pokrenuti novi proces za svaki skenirani port. U najgore slučaju možemo završiti sa oko 64000 otvorenih procesa po IP adresi. Ovo može uzrokovati veliko smanjenje performansi u radu računala. Iz tog razloga je nužno nositi se s ovim problemom. Dionaea se nosi s ovim problemom pomoću mehanizma *pomičnog prozora* TCP protokola.
- *rekurzivno samospajanje:* Pretpostavimo da postoji takav *shell* kod kojeg je potrebno izvršiti i koji će pokrenuti spajanje na *loopback* ili lokalnu adresu. U tom slučaju bi došlo do spajanja na slobodni port lokalnog računala, bio bi stvoren novi proces koji prihvaća konekciju (obavi *three-way handshake* do kraja) te krene stvarati zrcalnu konekciju. `SYN` paket zrcalnog zahtjeva će uzrokovati pokretanje novog procesa (jer dolazi na port lokalnog računala) koji će zapravo prihvatiti zrcalnu konekciju te će sam započeti novu zrcalnu konekciju. I na taj način „zrcaljenje“ se obavlja u beskonačnost. Iz tog razloga Dionaea provjerava je li „udaljeno računalo“ zapravo računalo s lokalnom adresom, te u tom slučaju odbacuje takve zahtjeve.

2.3 Honeyd

Honeyd je platforma za upravljanje s velikim brojem virtualnih honeypota. Obično se honeyd konfiguira da upravlja nealociranim IP adresama na postojećoj mreži. Za svaku IP adresu možemo reći honeydu kakvo ponašanje očekujemo od pripadajućeg virtualnog honeypota. Na primjer, mogli bismo postaviti virtualni web poslužitelj za kojeg izgleda kao da se vrti na Linuxu i sluša na portu 80. Možemo postaviti drugi virtualni honeypot za koji izgleda kao da su svi portovi otvoreni te se servisi vrte na Windows operacijskom sustavu. Pomoću honeyda se mogu stvarati ruting topologije koje se sastoje od stotine mreža i tisuće računala unutar jednog fizičkog računala.

2.3.1 Simuliranje virtualnih honeypota

Glavna prednost honeyda u odnosu na ostale honeypote je mogućnost konfiguriranja više virtualnih honeypota na jednom fizičkom računalu. Napadač može komunicirati sa svakim virtualnim honeypotom posebno i dobiti dojam da se radi o fizičkom računalu koje vrti određeni operacijski sustav, te su pokrenuti određeni servisi. Honeyd pokreće virtualne honeypote preko više IP adresa i fizičkih računala u isto vrijeme. Ovo svojstvo mu dozvoljava da se mreža popuni sa puno virtualnih honeypota koji su različito konfigurirani. K tomu je još moguće simulirati različite mrežne topologije. Simuliranje mrežnih topologija preko više fizičkih računala honeyd ostvaruje pomoću GRE tuneliranja.

Mrežne postavke

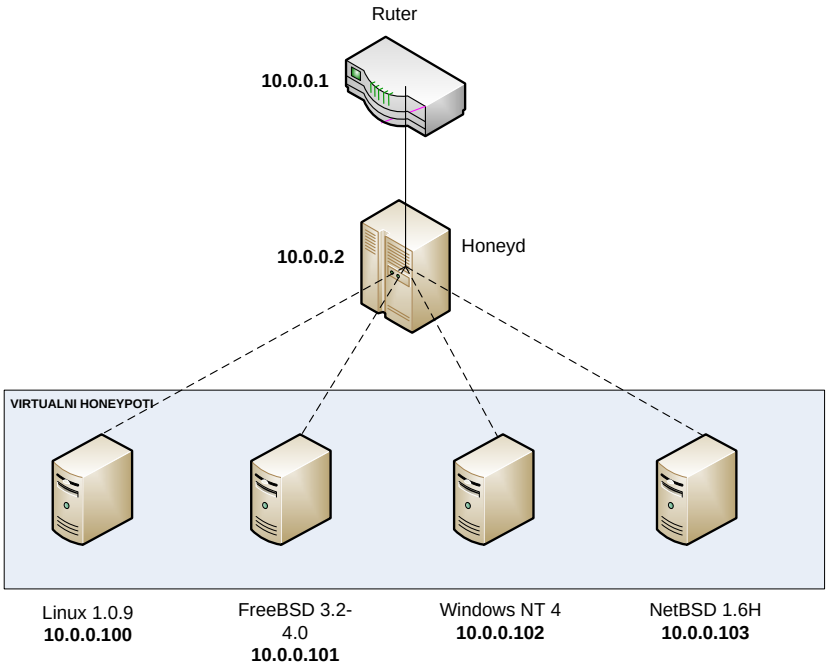
Da bi ostvarili opisanu konfiguraciju više virtualnih honeypota na jednom fizičkom računalu potrebno je napraviti određene mrežne postavke tako da paketi mogu doći do virtualnih honeypota. To je moguće ostvariti na dva različita načina:

Prvi način je *Proxy ARP*. Neka naš ruter ima adresu 10.0.0.1, a adresa honeyd fizičkog računala je 10.0.0.2 (Slika 1.). IP adrese virtualnih honeypota također leže u lokalnoj mreži i imaju adrese 10.0.0.100, 10.0.0.101, 10.0.0.102, 10.0.0.103. Kada napadač pošalje paket prema jednoj od IP adresa virtualnih honeypota, naš ruter će na temelju ruting tablica zaključiti da taj paket treba proslijediti mreži 10.0.0.0/24 s kojom je on direktno spojen. Da bi proslijedio paket bilo kojoj IP adresi u mreži potrebna mu je MAC adresa pripadajućeg sučelja. MAC adresa sučelja se dobiva pomoću ARP protokola gdje ruter šalje ARP zahtjev za MAC adresom određene IP adrese i pripadajuće sučelje odgovara ARP odgovorom. Pošto su adrese 10.0.0.100/30 pridružene virtualnim honeypotima koji su zapravo apstrakcija honeyda i samim time nemaju pridruženo fizičko sučelje, ARP odgovori ne dolaze ruteru, jer zapravo honeypoti ne vide ARP zahtjev. Potrebno je reći fizičkom sučelju Honeyd računala da prihvaća pakete namijenjene virtualnim honeypotima. Ako je MAC adresa pripadajuće 10.0.0.2 adrese Honeyd računala `1A:1A:60:F0:19:07`, onda je potrebno izvršiti sljedeću naredbu ako želimo da virtualni honeypot sa adresom 10.0.0.101 prima pakete:

```
/usr/sbin/arp -s 10.0.0.101 1A:1A:60:F0:19:07
```

Drugi način je stvaranje dodatnih ruta na ruteru 10.0.0.1. Ovaj način je moguć jedino ako imamo administrativnu kontrolu nad ruterom. Potrebno je postaviti statičku rutu koja sav promet za adrese 10.0.0.100/30 usmjerava na adresu 10.0.0.2.

```
route -n add -net 10.0.0.100/30 10.0.0.2
```



Simuliranje standardnog mrežnog stoga operacijskih sustava: Ovaj dodatak omogućuju honeydu da prevari *OS fingerprinting* Nmap i Xprobe alate tako da im izgleda kao da honeypot vrti određeni operacijski sustav. Dodatno, može se upravljati politikama slaganja fragmenata i skeniranja FIN zastavica. Honeyd koristi istu bazu podataka koju koriste i alati za *fingerprinting*. Kada honeyd treba poslati paket na mrežu, modificra ga pomoću baze podataka s obzirom na konfigurirani operacijski sustav.

Simuliranje proizvoljne topologije rutiranja: Honeyd može simulirati topologiju rutiranja podešenu unutar jednog fizičkog računala. Moguće je postaviti kašnjenja, gubitak paketa i karakteristike propusnosti. Za potrebe balansiranja opterećenja moguće je rasporediti ruting topologiju preko više fizičkih računala. Ovakva distribucija se postiže pomoću GRE tuneliranja. Napadaču koji pokušava skenirati mrežu izgleda kao da je topologija stvarna.

2.3.2 Simuliranje servisa

Skripte: Honeypot ne bi služio svojoj svrsi bez servisa s kojima napadač može „razgovarati“. Servisi nam pružaju potencijalno dodatne informacije o napadaču. U najjednostavnijoj izvedbi servis je aplikacija koja čita podatke sa standardnog ulaza (*stdin*) i piše na standardni izlaz (*stdout*). Ovakav servis je najlakše ostvariti pomoću *shell* skripte. Tako na primjer servis koji šalje poruku „Dobar dan!“, a zatim samo vraća sve što mu je poslano, bi u *shell* kodu izgledao ovako:

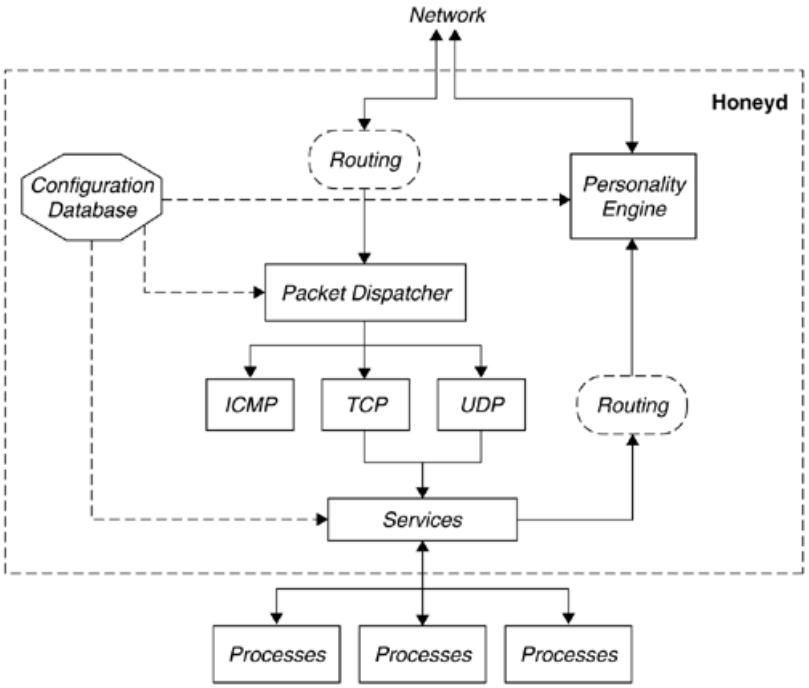
```
#!/bin/sh
echo „Dobar dan!“
while read dana
do
   echo „$data“
done
```

Ovu skriptu spremljenu u datoteku pod nazivom `dobardan.sh` potrebno je prebaciti u honeydev `scripts` direktorij i kroz konfiguracijsku datoteku odrediti na koji port honeyd treba povezati ovaj servis, npr. TCP port 23. Sa *shell* skriptama treba biti naročito oprezan zbog mogućeg *command injection* napada.

Python skripte: Python skripte su odvojene kao posebni mehanizam ostvarivanja servisa, jer osim korištenja standardnog ulaza i izlaza za interakciju sa mrežnim stogom, u Python jeziku se pomoću posebnog modul `honeyd` mogu implementirati neblokirajući ulazno/izlazni pozivi prema mrežnom sučelju. Potrebno je definirati posebne *callback* funkcije kako bi primali ili slali podatke na mrežu. Pošto su funkcije *callback*, održavanje stanja varijabli se provodi kroz argument koji primamo ili šaljemo u *callback* funkcijama. Za svaku konekciju se stvara posebni argument – *state object*.

- **honeyd_init(data)** – Ova funkcija se poziva kad je nova konekcija uspostavljana sa servisom. Zadatak ove funkcije je da vrati objekt u kojemu su upisane sve informacije koje su bitne za servis (*state object*).
- **honeyd_readdata(mydata, data)** – Ova funkcija se poziva kada su pristigli novi podaci za servis. Treba vratiti 0 za uspjeh, ili -1 za neuspjeh.
- **honeyd_writedata(mydata)** – Ova funkcija se poziva kadgod je mreža spremna da pošalje podatke na udaljeno računalo.
- **honeyd_end(mydata)** – Ova funkcija se poziva kada je konekcija prekinuta. U ovoj funkciji bi trebali osloboditi *state object*.

***Podsustavi*:** Problem s skriptama koje koriste standardni ulaz/izlaz je u tome što se za svaku konekciju stvara novi proces koji ju obrađuje (npr. u slučaju *shell* skripti se pokreće nova instanca *shell* interpretera). Honeyd omogućuje drugi način pokretanja servisa koji ima puno bolje performanse. Ovi servisi se nazivaju *podsustavi* zato što se konstantno vrte u pozadini i mogu primati više konekcija odjednom. Podsustavi su zapravo regularne Unix aplikacije koje pokreće honeyd. To znači da ne trebamo pisati svoju vlastitu emulaciju servisa, nego možemo iskoristiti gotovu aplikaciju. Pomoću podsustava se mogu implementirati složeniji protokoli. FTP je primjer protokola koji ne može biti implementiran pomoću skripti. Razlog tomu je što FTP koristi kanal za protok podataka koji je odvojen od kanala za kontrolne poruke. Kada FTP klijent pošalje poruku `get`, FTP poslužitelj dinamički otvara novi port za protok podataka. Servisi pisani kroz skripte ne mogu zahtijevati od honeyda otvaranje novih portova, niti mogu mijenjati konfiguracijsku datoteku tokom rada.



Konfiguracijske datoteke: Honeyd koristi tekstualnu datoteku kako bi se specificiralo na kojim IP adresama su dostupni određeni servisi, te koji standardni mrežni stog se emulira za svaki virtualni honeypot posebno. Može se specificirati da honeyd radi prosljeđivanje konekcije drugim računalima ili radi pasivni *fingerprinting*. Sve konfiguracije se izvode pomoću honeydevog konfiguracijskog jezika. U nastavku su opisani osnovne naredbe jezika:

- **create** – naredbom *create* se stvara novi *template*, tj. jedna „instanca konfiguracije“ koja se može iskoristiti za konfiguriranje virtualnog honeypota.
- **set** – naredba *set* postavlja osobnost (*personality*) iz Nmap *fingerprint* datoteke određenom *templateu*. Osobnost određuje ponašanje mrežnog stoga određenog operacijskog sustava. Moguće je postaviti podrazumijevanu akciju za servise koji se ne emuliraju, vrijeme rada računala (*uptime*), MAC adresu virtualnog honeypota.
- **add** – naredbom *add* dodajemo servis koji želimo emulirati. Potrebno je specificirati koji transmisijski protokol koristi servis i na kojem portu se vrti, pokreće li se kao podsustav ili skripta i kojom komandom se izvršava.
- **bind** – naredba *bind* je potrebna da bi finalizirali konfiguraciju virtualnog honeypota. Koristi se da bi *templateu* pridružili IP adresu. Ako paket dođe za određenu IP adresu koja nema pridružen *template*, honeyd koristi podrazumjevani *template*.
- **delete** – naredba *delete* se koristi za rekonfiguraciju virtualnog honeypota tijekom rada. Može se koristiti za brisanje *templatea* u potpunosti ili za brisanje određenog servisa iz *templatea*.

2.4 Argos

Argos je honeypot visokog stupnja interakcije koji je sposoban otkrivati nove napade (*zero-day attacks*). Koristi se tehnikom koja se zove *dynamic taint analysis* kako bi promatrao honeypot. Ovo je tehnika u kojoj su svi podaci primljeni preko mreže *označeni*. Označeni podaci se prate kroz memoriju u smislu da se sve varijable na koje ovi podaci utječu također označavaju. Ako u jednom trenutku ovi označeni podaci djeluju na tok izvršavanja (npr. naredba `JMP` na x86 arhitekturi procesora), tada Argos generira presliku memorije (*memory footprint*) nakon čega možemo započeti forenzičku analizu i otkriti ranjivost.

2.4.1 QEMU

Argos radi skupa sa QEMU procesor emulatorom. QEMU je procesor emulator koji koristi dinamičku translaciju instrukcijskog skupa naredbi kako bi emulirao procesor na kojemu se vrti gostujući operacijski sustav. QEMU nudi i jezgreni akcelerator Linux jezgru koji se zove KQEMU. Ovaj akcelerator omogućuje QEMU da x86 ili x86-64 strojni kod gostujućeg operacijskog sustava izvrši direktno na procesoru računala (koji mora imati istu arhitekturu) i na taj način ubrza virtualizacijsku okolinu. Instalacija Argosa se provodi uz instalaciju QEMU-a.

2.4.2 Kontrolni socket

Interakcija sa Argosom se izvodi preko kontrolnog socketa. Ovaj dodatak nam omogućuje da dobijemo trenutni status od Argosa ili pošaljemo komandu honeypotu. Također nam omogućuje da kontroliramo naš honeypot s udaljenog računala. Kontrolni socketom se pristupa Argosu na TCP portu 1347. Kada se spojimo na ovaj port primamo dvije vrste informacija:

1. `[ARGOS] Attack detected, code <3_letter_alert_description>` - obavještava nas da je Argos detektirao napad, npr. `code JMP`.
2. `[ARGOS] Log generated <argos.csi.random_id>` - obavještava nas da je generirana preslika memorije za kasniju analizu.

Komande koje možemo poslati Argosu na kontrolni socket omogućavaju automatizaciju procesa rada honeypota.

- `RESET`: virtualna mašina se resetira
- `SHUTDOWN`: virtualna mašina se gasi
- `PAUSE`: virtualna mašina se pauzira i svi procesi se pauziraju
- `RESUME`: virtualna mašina nastavlja svoj rad iz zaustavljenog stanja

3 Klijentski honeypot

Od 2007. godine u svijetu računalne sigurnosti dolazi do promjene u načinu širenja zlonamjernog koda i infekcije žrtava. Prije se malvare širio gotovo potpuno automatski, a ključan čimbenik koji je omogućavao takvo širenje bile su ranjivosti u operacijskim sustavima i mrežnim servisima.

Malver je s jednog zaraženog računala pretraživao mrežu za drugim računalima i potom se pokušao proširiti na njih. Kada bi uspio, istu radnju ponavljao bi na novozaraženom računalu. To je omogućavalo malveru da dostigne gotovo eksponencijalnu razinu širenja i u kratkom vremenu zarazi veliki broj računala bez ikakve interakcije s korisnikom.

U to vrijeme su se često pojavljivali različiti crvi koji bi nerjetko privukli i medijsku pozornost. Obično je kod njih najzanimljiviji podatak bio broj računala koji su uspjeli zaraziti. Neki od najpoznatijih crva su:

- Code Red – širio se 2001. godine iskorištavajući ranjivost u IIS web poslužitelju
- Welchia – širio se 2003. godine, iskorištava ranjivost u Microsoft RPC mehanizmu
- Blaster – slično kao i Welchia, širi se iskorištavajući ranjivost u Microsoft RPC mehanizmu
- SQL Slammer – Širio se iskorištavajući ranjivosti u MS SQL Server sustavu. U prvih 10 minuta je uspio zaraziti 75 000 računala.

Kako bi doskočili ovakvoj vrsti malvera sigurnosni stručnjaci su razvili honeypotove koji su opisani u prvom djelu ovog dokumenta. Oni bi emulirali ranjive servise i čekali da ih malver pokuša iskoristiti. Takvi honeypotovi su dobro funkcionirali i istraživačima bi omogućavali uvid i nadzor nad novim vrstama malvera i načinima njihova širenja.

Danas (2012. godine) situacija se iz temelja promijenila. Iako malver još uvijek postoji, on ima potpuno nove načine širenja i drukčiju ulogu. Napadači su se okrenuli klijentskim aplikacijama i korisnicima. Danas je teško pronaći malver koji se širi potpuno automatizirano putem mreže, bez interakcije korisnika. Gotovo svaki malware vezan je uz web i to je osnovna platforma za njegovo širenje. Neki od razloga za to su:

- Veliki rast popularnost web-a u zadnjih 5 godina. Web je postao osnovna platforma razmjene informacija i usluga na Internetu. Mnogim ljudima Internet i web su sinonimi.
- Korištenje weba kao platforme za poslovanje, u prvom redu kupovina i internet bankarstvo. Ovaj i razlog naveden prije, doveli su do velikog broja korisnika koji svoje financijske transakcije obavljaju putem interneta. To je zainteresiralo i kriminalce jer tu vide prostor za brzu zaradu.
- Poboljšana sigurnost operacijskih sustava. Proizvođači OS-ova su unaprijedili proces razvoja i poboljšali kvalitetu operacijskih sustava.
- Niska kvaliteta web aplikacija. Na webu postoji mnoštvo različitih vrsta aplikacija koje su razvijali različiti proizvođači ili pojedinci. Često te aplikacije nemaju zadovoljavajuću razinu kvalitete, ali se ipak koriste u produkciji. Kriminalci ovo iskorištavaju kako bi ranjive web aplikacije koristili za šrenje svog malvera.

Većina malvera danas se koncentrira na iskorištavanje ranjivosti u klijentskim aplikacijama koje su usko vezane uz web. U prvom redu to su web preglednici, zatim različiti preglednici

dokumanta (PDF, Word i sl.), a iskorištavaju se i ranjivosti u aplikacijama za pregled multimedijalnih sadržaja (npr. Flash).

Iskorištavanje ranjivosti odvija se putem web stranice. Ona sadrži zlonamjernu skriptu ili dokument koji, kada ga korisnik otvori, iskorištava ranjivost u njegovoj klijentskoj aplikaciji. Iskorištavanjem ranjivosti napadač dobiva kontrolu nad računalom korisnika i na njega postavlja svoj malver.

Takvi napadi zovu se drive-by-download napadi. Njihova ključna odlika je da korisnik mora posjetiti zlonamjernu wbe stranicu kako bi ranjivost bila iskorištena. Postoji mnogo različitih tehnika i načina pomoću kojih se korisnika može dovesti na zlonamjernu stranicu. Njihov opis je izvan opsega ovog dokumenta. Dovoljno je napomenuti da postoje dvije vrste drive-by-download napada:

1. Napad koji zahtjeva interakciju žrtve. Kod ovih napada žrtva mora dati svoj pristanak (najčešće je dovoljan pritisak na jednu tipku) kako bi instalirala malvare na svoje računalo. Napadači se pritom trude malver prikazali kao legitiman program i time prevariti potencijalnu žrtvu.
2. Napad koji ne zahtjeva interakciju žrtve. Ranjivost u klijentskoj aplikaciji žrtve je takva (tzv. code execution ranjivosti) da se malvare može automatski instalirati na njezino računalo čim korisnik posjeti neku zlonamjernu stranicu.

Klasični poslužiteljski honeypoti ne mogu prepoznati ni uhvatiti malver koji se širi na ovaj način. Zbog toga su razvijeni klijentski honeypotovi. Riječ je o specijaliziranim honeypotovima koji mogu pretraživati web i pri tome tražiti zlonamjerne web stranice i aplikacije.

Osnovna razlika između njih i klasičnih poslužiteljskih honeypotova je u pristupu. Dok poslužiteljski honeypotovi čekaju da malvare pokuša doći do njih, klijentski honeypotovi imaju proaktivnu ulogu i sami traže malvare.

Ključnu ulogu ima web. Klijentski honeypotovi optimizirani su za pretraživanje web-a. Oni automatski posjećuju web stranice i svaku posjećenu stranicu provjeravaju. Ukoliko jedna od stranica sadrži malver oni će to zabilježiti.

Iako ovo u osnovi zvuči jednostanvo, implementacija klijentskih honeypota se suočava s velikim problemima. Web se sastoji od mnogo složenih tehnologija, a za uspješnu provjeru klijentski honeypotovi moraju poznavati dobar dio tih tehnologija. Točnije rečeno, klijentski honeypotovi moraju što uspješnije emulirati web browser kako bi stvorili realan dojam posjećivanja. Emulacija web preglednika podrazumjeva izgradnju podrške za parsing HTML jezika i izvršavanje JavaScript-a te implementaciju DOM-a.

Ovo su složeni programski zadaci i različiti klijentski honeypotovi su različito pristupili rješavanju ovog problema. Treutno je dostupno malo implementacija klijentskih honeypotova. Većina njih su projekti koji su napušteni i neodržavani. U nastavku dokumenta donosimo prikaz tri najnovija klijentska honeypota.

3.1 Thug

Thug je klijentski honeypot niske interakcije pisan u programskom jeziku python. Kao što se od klijentskog honeypota može očekivati, Thug je dizajniran da emulira web preglednik i automatizirano analizira njihov sadržaj. Konačni cilj mu je otkriti i emulirati zlonamjeran sadržaj pronađen na web poslužiteljima odnosno web stranicama.

Thug je vjerojatno najmlađi od svih klijentskih honeypota, projekt je održavan u trenutku pisanja ovog dokumenta (srpanj 2012). URL adresa ovog projekta je: https://github.com/buffer/thug

Thug je namijenjen izvršavanju na Linux operacijskom sustavu. Ima gotovo potpunu implementaciju DOM-a koja je kompatibilna s W3C DOM Core, HTML, Event i View specifikacijama. Također, Thug koristi Googleov V8 JavaScript engine za emulaciju JavaScript koda na web stranicama.

Važno je napomenuti da Thug može prepoznati shellcode unutar različitih zlonamjernih linkova koristeći libemu. Libemu je biblioteka otvorenog koda za emulaciju instrukcija na x86 platformi. Posebno je optimizirana za detekciju shellcoda.

Valja imati na umu da instalaciju Thug-a nije trivijalna budući da ovisi o mnogo različitih biblioteka i python modula. Osim već spomenutih libemu i V8 za rad Thuga potebno je instalirati i BeautifulSoup, Html5lib, MongoDb, Zope interface itd.

Nakon instalacije Thug honeyclient se pokreće iz komandne linije naredbom:

python thug.py [options] url

Thug podržava nekoliko opcija. Najvažnije su navedene u slijedećoj tablici:

Naziv opcije	Kratica	Opis
--output	-o	Datoteka u koju treba bilježiti log
--referer	-r	Određuje URL koji treba koristi za Referer zaglavlje u HTTP protokolu
--useragent	-u	Određuje što treba postaviti za User-Agent zaglavlje u HTTP protokolu
--adobepdf	-A	Određuje koji verziju Adobe PDF čitača Thug treba emulirati.

Thug ima sposobnost emulacije različitih verzija web preglednika kako bi uspješno pokrenuo iskorištavanje ranjivosti na stranici koju analizira. U Thugu se ovo nazivaju osobnosti (eng. personality). Trenutno su podržane sljedeće osobnosti:

- Internet Explorer 6.0 (Windows XP)
- Internet Explorer 6.1 (Windows XP)
- Internet Explorer 7.0 (Windows XP)
- Internet Explorer 8.0 (Windows XP)
- Internet Explorer 6.0 (Windows 2000)
- Internet Explorer 8.0 (Windows 2000)
- Safari 5.1.1. (MaxOS X 10.7.2)

Nakon što analizira određenu stranicu Thug će pohraniti rezultate analize u jedan direktorij. U njemu će se nalaziti i izvještaj analize (analysis.xml) datoteka i zabilježeni zlonamjerni programi. Izvještaj analize (datoteka analysis.xml) pisana je u MAEC formatu. Također, sve preuzete datoteke prilikom analize stranice spremaju se u direktorije temeljene na tipu sadržaja.

Još jedna mogućnost Thug-a je da rezultate analize pohranjuje u MnogoDB bazu podataka. Kako bi se koristila ova mogućnost treba instalirati python biblioteke za spomenutu bazu podataka. Thug će sam otkriti da li su potrebne biblioteke instalirane i ukoliko jesu uz pohranu rezultata analize u direktorij, koristit će i MongoDB.

Još dvije važne mogućnosti Thug-a koje treba naglasiti su podrška za preuzimanje web sadržaja preko proxy-a i podrška za analizu lokalno pohranjenih web stranica. Ovo potonje može posebno biti korisno istraživačima koji imaju primjerke web stranica pohranjene na lokalnom disku, a koje nisu dostupne putem mreže.

Najveći nedostatak Thuga je činjenica da nije namijenjen za automatsko pretraživanje web stranica. Odnosno Thug nema mogućnosti raditi crawling.

3.2 Phoneyc

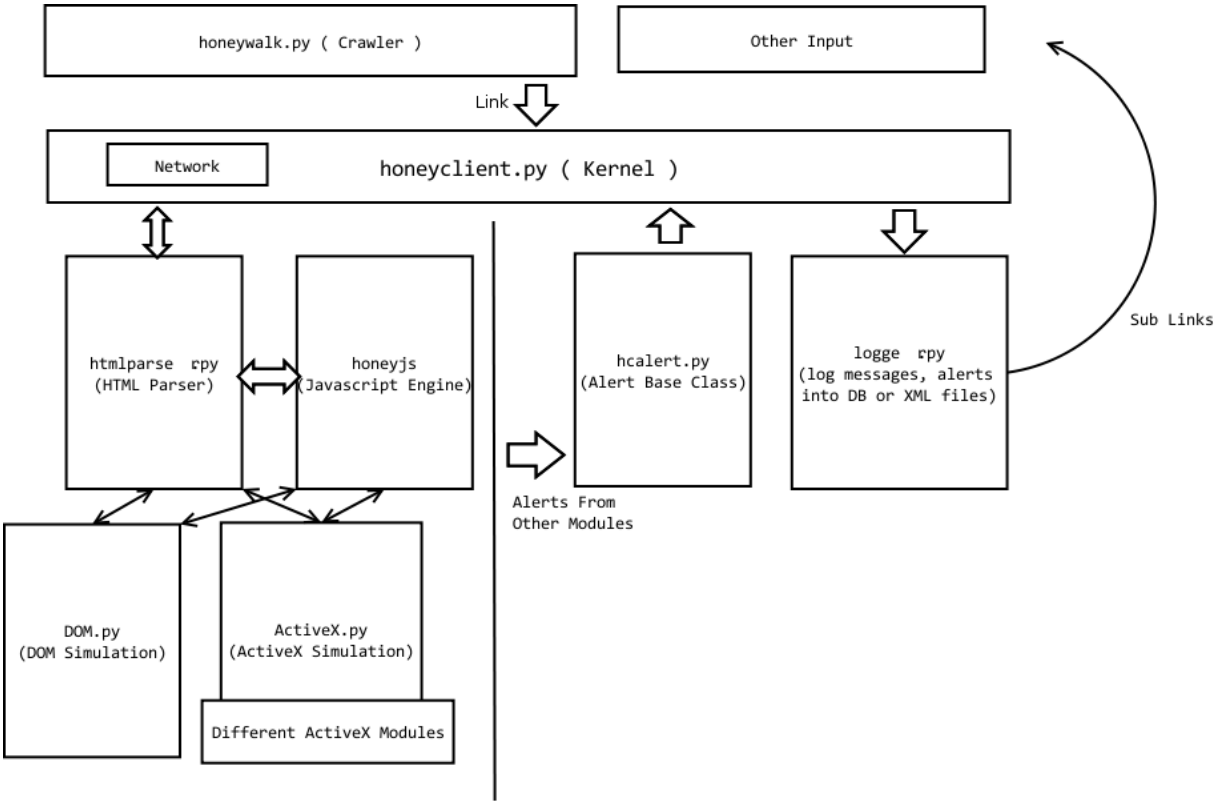
Phoneyc još je jedan klijentski honeypot niske interakcije. Za razliku od Thug-a Phoneyc je stariji i manje održavan. Web adresa ovog projekta je: http://code.google.com/p/phoneyc/

Slično kao i Thug, Phoneyc koristi emulaciju kako bi „odglumio“ različite ranjivosti koji se iskorištavaju protiv web preglednika. Pri tome Phoneyc imitira izgled preglednika prema pojedinoj web stranici i podržava parsing te izvršavanje skriptnog koda. Phoneyc je u potpunosti pisan u programskom jeziku Python, ali slično kao i Thug koristi vanjske module i biblioteke kako bi postigao određenu funkcionalnost.

Phoneyc razumije i može obraditi HTML jezik, također ima ugrađenu podršku za izvršavanje JavaScript koda putem monkeyspider biblioteke. Osim JavaScript Phoneyc podržava i kod pisan Visual Basic programskom jeziku. Phoneyc može emulacijom deobfuscirati skripte.

Što se detekcije ranjivosti i zlonamjernog koda tiče Phoneyc koristi dva različita pristupa. Odmah nakon preuzimanja sadržaja s udaljenog web servera Phoneyc će koristit ClamAV anti-virusni alat kako bi provjerio da li stranica ima kakav zlonamjeran sadržaj. Nakon toga Phoneyc pokreže parsing HTML sadržaja i iz njega izdvaja potencijalne skripte. Skripte se potom emuliraju i njihov sadržaj analizira. Ukoliko Phoneyc tada otkrije moguće iskorištavanje ranjivosti ili zlonamjeran kod, ispisat će upozoronje i zabilježiti URL u log datoteku.

Slijedeća slika (preuzeto iz dokumentacije) prikazuje arhitekturu Phoneyc honeypota.



Osnova cijelog Phoneyc klijenta je honeyclient.py – jezgra i server. On preuzima sadržaj s interneta, predaje ga na anlizu i ovisno o potrebi logira ili upozorava korisnika o pronađenim prijetnjama. Kao što je već spomenuto Phoneyc ima mogućnost parsiranja HTML dokuemnata i izvršavanja JavaScript koda. Ova obrada događa se u modulima htmlparse.py, DOM.py i honeyjs.

Jezgra Phoneyc honeypota (honeyclient.py) ulaze, odnosno web stranice koje treba provjeriti dobiva ručno ili putem crawlera (honeywalk.py). Postojanje crawlera još jedna je razlika između Thuga i Phoneyca. Crawler omogućuje automatizirano pretraživanje web sadržaja u potrazi za linkovima. Dok se Thugu URL za analizu mora zadati ručno, Phoneyc može analizom jedne web stranice otkriti povezince na druga web mjesta i potom njih analizirati. Ovakvo ponašanje puno je bliže ponašanju stvarnih web preglednika.

Pokretanje Phoneyc programa je relativno jednostavno. Postoji samo nekoliko važnijih opcija koje su navedene u slijedećoj tablici:

Naziv opcije	Kratica	Opis
--logfile	-l	Datoteka u koju treba bilježiti log
--useragent	-u	Određuje što treba postaviti za User-Agent zaglavlje u HTTP protokolu
--replace-nonascii	-n	Zamjena svih non ASCII znakova znakom razmaka u HTML i JS dokumentima.

3.3 HNS Capture-HPC NG

HNS je kratica za HoneySpider Network Capture. Riječ je o zajedničkom projektu između poljskog i nizozemskog CERT-a te SURFNeta. Rezultat tog projekta je klijentski honeypot visoke interakcije nazvan HNS Capture-HPC NG.

Capture-HPC NG je nasljednik starijeg honeypota: Capture-HPC. Ovaj projekt je proširio stariju verziju i dodao mu nove mogućnosti. Sam honeypot dostupan je na web stranici: http://pl.honeynet.org/HoneySpiderNetworkCapture

Nažalost dokumentacija za Capture-HPC NG je slaba i gotovo da nema jasnih uputa za upotrebu. Dostupni su samo kratke upute za instalaciju. Capture-HPC NG je klijentski honeypot visoke interakcije. On ne emulira preglednike i njihove ranjivosti već za surfanje internetom koristi stvarne web preglednike. Takav honeypot predstavlja znatno veću opasnost budući da napadač može kompromitirati računalo koje se koristi u honeypotu i preuzeti kontrolu nad njim.

Kako bi izbjagao takav scenarij Capture-HPC NG koristi virtualna računala. Naime, Capture-HPC NG se sastoji od dva dijela. Prvi dio je poslužitelj koji kreira i upravlja virtualnim računalima te njih koristi za analizu zlonamjernog koda. Drugi dio je klijent koji se instalira u virtualno računalo. On komunicira s poslužiteljem kako bi prikupio nove radne zadatke i upravlja samim preglednicima.

Klijent upućuje web preglednik na sumnjivo web mjesto i potom prati promjene na sustavu. Ukoliko otkrije da se nakon posjeta određenom web mjestu na sustavu nalazi zlonamjerni kod zaključuje da je riječ o zlonamjernoj web stranici. Nakon analize, poslužitelj vraća virtualno računalo u izvorno stanje koje nije inficirano. Capture-HPC NG honeypot koristi crawler kako bi automatski analizirao različita web mjesta.

Poslužitelj je dizajniran za Linux operacijski sustav, a svaki klijent za Windows XP operacijski sustav. Klijenti koriste Internet Explorer 6.0 kako bi testirali web stranice na ranjivosti.

Ovakav pristup s korištenjem virtualnih računala i visoke interakcije donosi veliku prednost u analizi ranjivosti i zlonamjernog koda koji se koristi na web-u. Capture-HPC NG može otkriti nove i nepoznate ranjivosti. Cijeli sustav predstavlja realnu okolinu i puno je veća vjerojatnost da će ranjivost u preglednicima biti iskorištena te da će honeypot uspjeti uhvatiti malvare.

Mane ovog pristupa su složena instalacija i konfiguracija te znatno veća opasnost od kompromitiranja i otkrivanja honeypota. Zbog toga je prilikom instalacije ovakvog honeypota potrebno dodatnu pažnju posvetiti sigurnosti.

4 Literatura

1. http://nepenthes.carnivore.it/
2. http://sourceforge.net/projects/honeytrap/
3. http://dionaea.carnivore.it/
4. http://sourceforge.net/projects/amunhoney
5. http://buffer.github.com/thug/
6. http://pl.honeynet.org/HoneySpiderNetworkCapture
7. http://code.google.com/p/phoneyc/