UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Simon Mavsar

PROTOTIP RAZISKOVALNEGA MUHOLOVCA

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Ljubljana, julij 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00424/2009
Datum: 15.01.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **SIMON MAVSAR**

Naslov: **PROTOTIP RAZISKOVALNEGA MUHOLOVCA**
RESEARCH HONEYPOT - A PROTOTYPE

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Opišite problematiko napadov in vdorov v računalniške sisteme. Napade klasificirajte in opišite teoretično ozadje. Analizirajte možnosti za zaznavanje in preprečevanje napadov ter zlasti orodja, ki so danes na voljo na tem področju. Nato se osredotočite na koncept muholovca, preučite prednosti in slabosti ter predlagajte lastno zasnovo muholovca. Predlog implementirajte, preizkusite v praksi ter nazadnje kritično ocenite izvedbo in uporabnost.

Mentor:
M. Ciglarič
doc. dr. Mojca Ciglarič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Franc Solina

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Simon Mavsar

PROTOTIP RAZISKOVALNEGA MUHOLOVCA

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Mojca Ciglarič

Ljubljana, julij 2009

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a ___SIMON MAVSAR_____,

z vpisno številko ___63990269_____,

sem avtor/-ica diplomskega dela z naslovom:

_PROTOTIP RAZISKOVALNEGA MUHOLOVCA_____

_RESEARCH HONEYPOT – A PROTOTYPE_____

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  ___MOJCA CIGLARIČ_____

  in somentorstvom (naziv, ime in priimek)

  ___ ANDREJ KREVL _____

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne __6.7.2009_____Podpis avtorja/-ice:
_____

ZAHVALA

Na prvem mestu se bi rad zahvalil as. Andreju Krevl in doc. dr. Mojci Ciglarič za njihovo pomoč med izvajanjem in pisanjem diplomske naloge.

Še posebej gre zahvala as. Andreju Krevl, ki je kot nadzornik računalniškega omrežja na srednji šoli Trbovlje omogočil uporabo ločenega dela omenjenega omrežja za izvedbo eksperimenta.

Zahvala tudi moji družini, prijateljem in še posebej mojemu dekletu Sandri za izkazano podporo, bodrenje in prenašanje mojih *muh* med pisanjem diplomske naloge.

KAZALO

SLOVAR, KRATICE IN AKRONIMI

Termin	Definicija
Muholovec (ang. Honeypot)	Muholovec je računalniški sistem, postavljen kot past, ki je zasnovana tako, da privabi in beleži neavtorizirano aktivnost na danem računalniškem sistemu
Sladki kolač (ang. Honeytoken)	Lažna elektronska enitiiteta, postavljena kot past z namenom da privabi in beleži neavtorizirane dostope
Simx	Prototipna verzija visoko interaktivnega muholovca
Alarm	Opozorilni signal, ki sporoča morebitni napad na opazovani sistem
Lažni alarm	Alarm sprožen na osnovi legalnega prometa, kateri ne predstavlja grožnje
Filtriranje alarmov	Proces kategorizacije zajetih alarmov z namenom izločiti lažne
Lažni negativ	Nesposobnost sistema za detekcijo vdorov da zazna dejanski vdor na opazovanem sistemu
Šum	Avtoriziran (ali legalen) promet, ki sproži lažni alarm
Neizkušen napadalec (ang. script kiddie)	V računalniškem žargonu poimenovan tehnično podkovan vdiralec, kateri uporablja že napisana orodja.
Črv (ang. worm)	Računalniški črv je program z spodobnostjo samotojne replikacije.
Virus	Računalniški virus je programska koda, ki se je sposobna razmnoževati in prenašati v računalniku brez vednosti in volje uporabnika. Gostitelj virusa je računalniški program oziroma izvršna datoteka.
Usmerjevalnik (ang. router)	Usmerjevalnik je naprava, ki povezuje dve ali več različnih omrežij. Njegove funkcije so omejevanje prometa, prenašanje prometa na manjša omrežja in izbira najustreznejše poti za potovanje podatkovnih paketov do njihovega cilja.
Sistem za odkrivanje napadov	Programska oprema za pridobivanje podatkov o okolju, potrebnih za analizo obnašanja sistema in odkrivanje varnostnih lukenj, poskusov vdorov, odprtih ranljivosti, ki bi lahko pripeljale do potencialnih vdorov.
Požarni zid (ang. firewall)	Požárni zíd je verjetno najpogostejši varnostni izdelek s področja omrežne varnosti. Potrebuje ga že skoraj vsaka naprava, povezana v internet. Požarni zidovi so namenjeni ločevanju dveh odsekov omrežij, pogosto enemu odseku zaupamo, drugemu pa ne.

Kratica	Definicija
RFC	Zahteva za mnenja (ang. Request for Comments; kratica RFC) je dokument, ki določa tehnične vidike Interneta. V začetku so ti dokumenti služili za zbiranje informacij tehničnih udeležencev v omrežju. Veliko RFC-jev temu namenu služi še danes, številni pa so samo zapisi dejstev.
IP	IP (ang. Internet Protocol) je primarni protokolni gradnik, ki sestavlja internetni mrežni sklad. Zadolžen je za usmerjanje in dostavo (različnih - glede na transportne protokole na spodnje ležečih plasteh) mrežnih paketov od izvornega do ciljnega računalnika
ICMP	Protokol ICMP (ang. Internet Control Message Protocol) se uporablja, kot že ime pove, za pošiljanje nadzornih sporočil in sporočil stanja internet omrežja. Pakete ICMP razlikujemo po tipu sporočila, ki ga nosijo, to je lahko zahteva, odgovor na zahtevo, statusna informacija in vrsta napake.
TCP	TCP/IP (ang. TCP »Transmision Control Protocol«, protokol za nadzor prenosa, ter IP »Internet Protocol«, internetni protokol) ali Internetni sklad protokolov (angleško Internet protocol suite) je množica protokolov, ki izvaja protokolski sklad prek katerega teče internet.
UDP	UDP (ang. User Datagram Protocol) je nepovezovalni protokol za prenašanje paketov. Nepovezovalni pomeni, da odjemalec in strežnik ne vzpostavita povezave, ampak strežnik pošilja pakete odjemalcu in ne preverja, če je odjemalec pakete dobil.
SMTP	SMTP (ang. Simple mail transfer protocol) je preprost protokol za prenos elektronske pošte, ki je standard za prenos elektronske pošte na Internetu.
FTP	FTP (ang. File transfer protocol, »protokol za prenos datotek«) je programski standard za prenos datotek med računalniki z različnimi operacijskimi sistemi. Spada v aplikacijsko raven internetnega nabora protokolov.
DNS	DNS (ang. Domain Name System); globalno distribuirano omrežje strežnikov imen domen v internetu, ki izvajajo preslikavo med imenskimi in številčnimi naslovi IP. Sistem DNS hierarhično ureja prostor edinstvenih imen računalnikov v internetu.
PMTU	PMTU (ang. Path Maximum Transmition Unit) ali maksimalna vrednost velikosti mrežnega paketa na poti čez dano omrežje
VoIP	Voice over IP – internetna telefonija

NAT	Network Address Translation
WHOIS	Je na TCP zasnovan protokol, ki se uporablja za ugotavljanje lastnika domene ali naslova IP na internetu.

POVZETEK

Muholovec je računalniški sistem, postavljen kot past, ki je zasnovana tako, da privabi in beleži neavtorizirano aktivnost na danem računalniškem sistemu. Na grobo ga lahko opišemo kot računalniški sistem, dosegljiv preko mreže, na prvi pogled nazaščiten ter z mamljivo vsebino, v resnici pa sestavni del izoliranega ter skrbno nadzorovanega segmenta računalniškega omrežja.

Cilj diplomske naloge je pregled in predstavitev koncepta muholovec, kot ene izmed tehnik za varovanje računalniškega omrežja, ki nam pomaga bolje se zaščititi pred vdori ter razvoj prototipnega ogrodja za uporabo in nadaljevanje raziskovalnega dela na temo visoko interaktivnih muholovcev. Po definiciji so tovrstni tipi muholovcev vzdrževani iz strani prostovoljnih, neprofitnih raziskovalnih ali izobraževalnih ustanov, katere imajo za cilj zbirati informacije o motivih in taktikah mrežnih napadalcev. Tovrstni tipi muholovcev so uporabljeni zgolj kot sredstvo za zbiranje informacij o spletnih grožnjah. Zbrane informacije se lahko potem uporabi za izboljšanje zaščite pred le temi. Raziskovalni modeli muholovec so ponavadi zelo zahtevni za postavitev in vzdrževanje in kot taki največkrat uporabljeni iz strani raziskovalnih, vladnih in tudi vojaških organizacij.

V prvem delu naloge imamo predstavljen teoretični pregled delovanja omrežij, vdore vanje, tehnike zaščite ter predstavitev muholovec koncepta, kjer se spoznamo z idejo in motivi za razvoj tovrstnih orodij ter natančen pregled več tipov (tako komercialnih kot odprokodnih) muholovec orodij, kateri so nam trenutno na voljo. V nadaljevanju imamo kot praktični del diplomske naloge predstavljen razvoj visoko interaktivnega raziskovalnega muholovca, poimenovanega *simx*, katerga sem v nadaljevanju uporabil za konkretno raziskovalno delo na ločenem segmentu računalniške mreže srednje šole Trbovlje. Ker je muholovec po definiciji past, katere namen je privabiti in beležiti vso aktivnost na takem sistemu, se mi je pojavil praktičen problem, kako privabiti tovrstno aktivnost na opazovan sistem. S tem namenom je bila kot neke vrste poligon praktičnega dela naloge razvita spletna stran, katere namen je kot vaba privabiti organski, neavtomatiziran mrežni promet, v kontekstu katerega bi lahko prišlo tudi do napada na opazovani strežnik.

Z namenom ponazoritve dodane vrednosti uporabe muholovca, sem se odločil še za razširitev praktičnega dela naloge z rezultati uporabe standardnih orodij za detekcijo mrežnih napadov (IDS) na dveh dodatnih kontrolnih točkah, postavljenih v Ljubljani in Krškem. Na ta način mi je bila omogočena konkretna primerjava obeh omenjenih načinov (standardna orodja za detekcijo mrežnih napadov ter muholovec), in sicer z analizo zajetih podatkov o napadih na obeh opazovanih sistemih. Rezultati raziskovalnega dela so predstavljeni v zaključnem delu naloge.

Ključne besede

muholovec, omrežje muholovcev, sistem za odkrivanje mrežnih vdorov, računalniška varnost, odkrivanje vdorov, požarni zid

ABSTRACT

A honeypot is a trap set to detect attempts at unauthorized use of information systems. Generally it consists of a computer, data, or a network site that appears to be part of a network but which is actually isolated, (un)protected, and monitored, and which seems to contain information or a resource that would be of value to attackers.

Goal of research work was to evaluate and summarize various concepts of honeypot solutions as one of network security techniques available today. Additionally custom made high interaction honeypot solution was developed in context of this work, which was later used to produce some research results presented at the end. High interaction research honeypots are by definition run by a volunteer, non-profit research organization or an educational institution to gather information about the motives and tactics of the Blackhat community targeting different networks. These honeypots do not add direct value to a specific organization. Instead they are used to research the threats organizations face, and to learn how to better protect against those threats. This information is then used to protect against those threats. Research honeypots are complex to deploy and maintain, capture extensive information, and are used primarily by research, military, or government organizations.

In first part of paper I've prepared a theoretical overview of basic concepts of most widely used internet protocols, different ways to break and harden network security and detailed presentation of honeypot concepts. As practical part of research work I've developed my own version of high interaction honeypot solution, called *simx*, which was used for actual experiment on isolated subnet of academic network. Because honeypot is a trap set to detect unauthorized activity, very practical problem surfaced, and that is how to lure such activity on our honeynet. In order to overcome described limitation I've decided to prepare a web page which was designed to generate organic, non automated traffic which may produce higher volume of desired activity.

In order to present added value of honeypot deployment I've decided to extend practical part of this research work with results produced by standard intrusion detection (IDS) tools on two additional control nodes set up in Ljubljana and Krško. This approach made effective comparison (standards intrusion detection tools vs. honeypot solutions) possible by detailed analysis of captured material on all thee control nodes. Results of research work are presented in second, practical part of this paper.

Keywords

honeypot, honeynet, intrusion detection system, network security, intrusion detection, firewall

1 UVOD

Muholovci so relativno nova in zelo dinamična tehnologija. Ravno zaradi dinamičnosti nam predstavlja sama definicija termina *muholovec* nemalo problemov. Kot prvo in glavno razliko od običajnih varnostnih orodij velja omeniti, da muholovci niso rešitev sama po sebi in da brez nadaljne aktivnosti in analize zajetega materiala ne rešijo nobenega konkretnega varnostnega problema. Kar je v nasprotju z obstoječimi tehnologijami, kot so požarni zidovi ali sistemi za odkrivanje vdorov, pri katerih je definicija in samo razumevanje namenskosti dosti lažje ali celo samoumevno, saj so narejeni za reševanje točno zastavljenih ciljev ali problemov. Požarni zidovi so preventivna orodja, s katerimi ščitimo mrežne segmente pred zunanjim prometom. Sistemi za zaznavanje vdorov so alarmna orodja, katera zaznavajo in opozarjajo na neavtorizirano aktivnost, medtem ko so muholovci težje opredeljivi, saj jih je mogoče uporabiti v preventivne namene, zaznavanje, zajem podatkov ter še veliko več …

Po definiciji iz [1] lahko definiramo muholovca kot: *"Muholovec je informacijski sistem, katerega glavna vrednost je v neavtorizirani uporabi le tega."*

Iz te definicije ni razvidno delovanje niti namen muholovca, namreč definicija samo podaja pogled na samo vrednost te tehnologije. Torej, muholovci so tehnologija, katere vrednost leži v napadalčevi interakciji s takim sistemom. Iz tega lahko potegnemo glavni koncept, ki je skupen vsem muholovcem. Nobena aktivnost na takem sistemu ne sme biti legalnega izvora, ali povedano obratno, vse transakcije na tem sistemu so po definiciji neavtorizirane ali škodljive in kot take predmet nadaljne analize. Muholovci ne nudijo nobenega servisa ter nimajo nobene ciljne vrednosti poleg raziskovalne naloge. Kot primer uporabe muholovca navedimo namestitev na spletni ali datotečni strežnik, kateri se ponavadi nahaja v ločenem segmentu računalniškega omrežja in je namenjen izključno raziskovalnim namenom.

Če razširimo to idejo še naprej, za muholovca sploh ni nujno, da je računalniški sistem. Lahko je marsikatera digitalna entiteta, brez dejanske produkcijske vrednosti. Za ponazoritev navedimo bolnico, katera naredi lažno elektronsko kartoteko za pacienta z imenom Barrack Hussein Obama in to kot muholovec komponento, v žargonu imenovano tudi sladki kolač, nastavi v svojo bazo pacientov. Ker so tovrstne kartoteke muholovci, je samoumevno, da je marsikakšen dostop do le teh neavtoriziran in kot tak predmet nadaljne preiskave. Prav ta ohlapnost definicije koncepta muholovec je lahko njegova velika prednost (ali pa tudi pomanjkljivost).

1.1 DELOVANJE OMREŽJA

Preden preidemo na bistvo naloge, si poglejmo osnove delovanja interneta ter opis problema, s katerim se naloga ukvarja. Osnove so predstavljene samo do te mere, da so še relevantne za nadaljevanje naloge.

1.1.1 OSI referenčni model ter internetni sklad TCP/IP

Konec 60-tih ter v zgodnjih 70-tih letih prejšnjega stoletja se je z razmahom računalniške tehnologije pojavljala zahteva po povezovanju posameznih računalniških sistemov v računalniška omrežja. Ker pa je bilo dejstvo, da je vsak proizvajalec mrežne opreme imel svoje specifikacije ter ideje, kako to storiti, se je pojavila močna potreba po poenotenju ter definiranju protokolov računalniške komunikacije.

1.1.2 OSI referenčni model

OSI (Open System Interconnect) referenčni model je abstraktna predstavitev nivojske komunikacije med računalniškimi sistemi, povezanimi v mrežo. Razvit je bil kot del širše OSI iniciative – poizkus standardizacije in poenotenja komunikacije ter komunikacijskih vmesnikov med različnimi računalniškimi sistemi. Po tem modelu je mrežni sklad deljen na 7 slojev:

- **Aplikacijski**

  Aplikacijski nivo predstavlja aplikacija, katera je v direktni interakciji ali uporabi s strani končnega uporabnika. Primeri: klienti za branje in pošiljanje elektronske pošte, spletni brskalniki...

- **Predstavitveni**

  Predstavitveni nivo poskrbi za semantično pretvorbo podatkovnih tokov med obema sosednjima nivojema (v obe smeri). Glede na dejansko implementacijo internetnega mrežnega skalda ga uvrščamo v aplikacijski nivo le tega.

- **Sejni**

  Sejni nivo skrbi za nadzor in kontrolo dialoga med dvema ali več aplikacijami, katere tečejo vsaka na svojem računalniškem sistemu. Skrbi za postavljanje, vzdrževanje ter prekinitev dialoga. Glede na dejansko implementacijo internetnega mrežnega sklada tudi sejni nivo uvrščamo v aplikacijski nivo le tega.

- **Transportni**

  Transportni nivo poskrbi za transparetni prenos mrežnih paketov med obema točkama komunikacije ter s tem zagotavlja zanesljiv podatkovni tok za zgornje nivoje.

- **Mrežni**

  Mrežni nivo zagotavlja vse potrebno (usmerjanje, fragmentacija, defragmentacija, ugotavljanje napak) za zanesljiv ter čim bolj optimalen prenos (različnih) mrežnih paketov med različnimi topologijami računalniškega omrežja.

- **Povezovalni**

  Povezovalni nivo poskrbi za podatkovne prenose med različnimi mrežnimi entitetami ter zaznava in odpravlja morebitne napake, ki se pojavijo na tem nivoju.

- **Fizični**

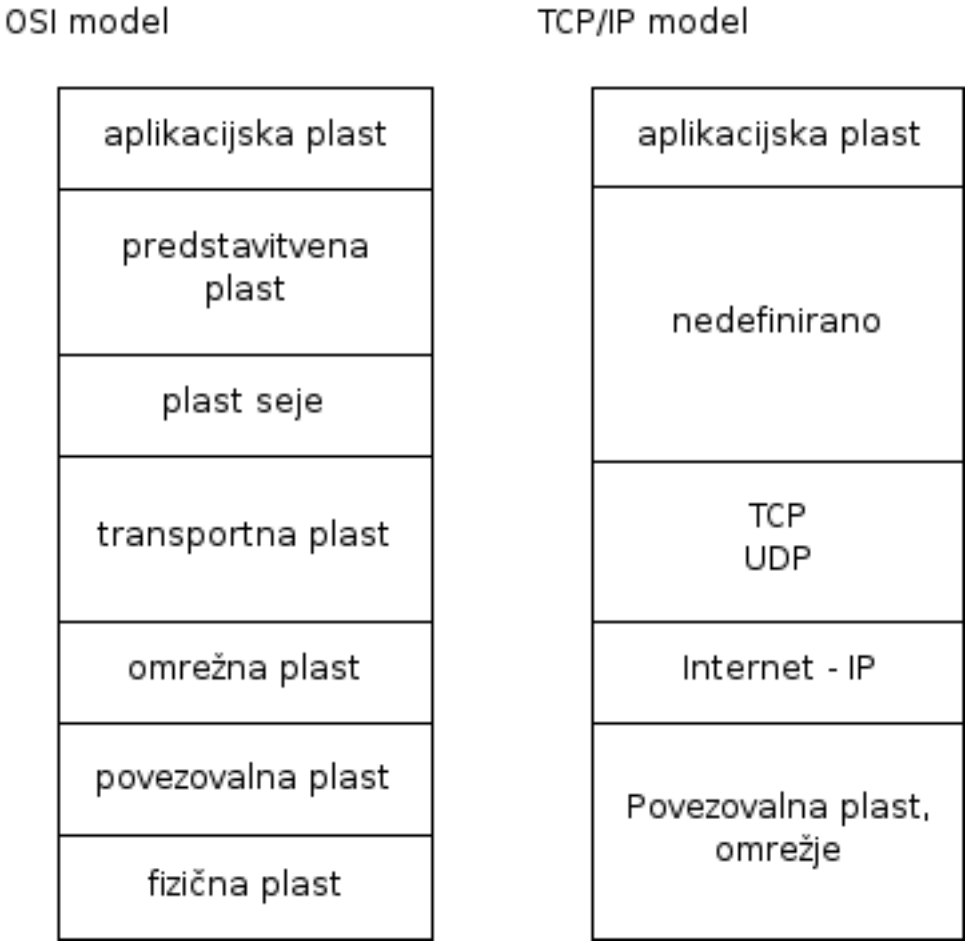
  Fizični nivo definira električne in fizične karakteristike (napetostni nivoji, konektorji, mrežne kartice, usmerjevalniki …) ter lastnosti mrežnih naprav.

1.1.3 Internetni mrežni sklad

Medtem ko je OSI referenčni model neke vrste abstrakcija mrežnega sklada na papirju, lahko rečemo tudi *de iure* standard ali predlog, kako naj bi idealen mrežni sklad bil sestavljen, se je v praksi bolj uveljavil internetni mrežni sklad, kot *de facto* standard, katerega lahko razdelimo na 4 nivoje.

Diagram: Primerjava obeh (OSI, TCP/IP) modelov

Za boljšo ponazoritev nam spodnji diagram dobro povzame primerjavo obeh omenjenih modelov.



Slika 1. Primerjava OSI in TCP/IP modela.

1.1.4 Aplikacijska plast (HTTP, FTP, …)

Aplikacijska plast predstavlja najvišji nivo mrežnega sklada, to je samo aplikacijo, za katero je končni uporabnik ter s tem povezane aplikacijske protokole, kot so: SMTP/POP3, FTP, HTTP …

1.1.5 Transportna plast (TCP/UDP/SCTP)

Transportna plast zagotavlja zanesljiv prenos mrežnega paketa, odpravljanje napak, kontrolo prenosa in fragmentacijo, neodvisno od spodnje ležečih topologij omrežja.

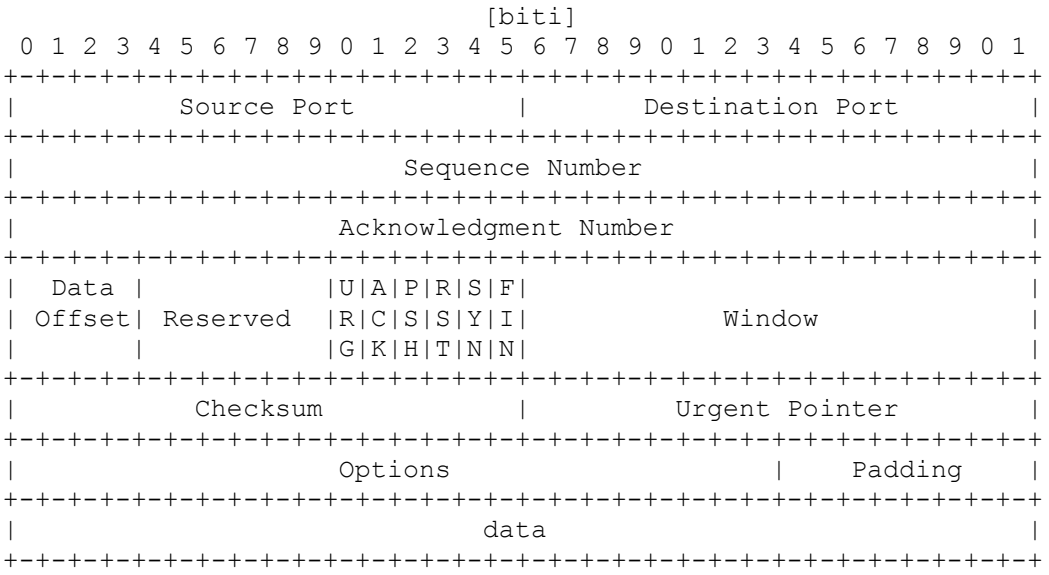
1.1.5.1 TCP (Transport Control Protocol)

Transport Control Protocol je eden izmed ključnih protokolnih gradnikov, ki sestavljajo internetni mrežni sklad. TCP z Internet Protocol (IP) sestavlja ali zaokrožuje jedro internetnega sklada, katerega pogosto imenujemo kar TCP/IP. Medtem ko je naloga IP nivoja, da usmerja internetne pakete od izvornega računalniškega sistema do ponora, nam TCP nivo služi za samo kontrolo prenosa med tema dvema točkama (npr. med spletnim strežnikom na eni strani ter spletnim odjemalcem na drugi). TCP nam torej zagotavlja zanesljiv in urejen (v pravilnem zaporedju) prenos podatkovnih nizov med dvema

računalniškima sistemoma. Poleg tega TCP skrbi še za velikosti mrežnih paketov, za usklajevanje hitrosti oddajanja in sprejemanja itd.

TCP zaglavje je vgnezdeno znotraj IP paketa ter sledi IP zaglavju, katero vsebuje informacijo o tipu transportnega protokola. Tako imamo zagotovljeno podporo za uporabo še ostalih transportnih protokolov.

Sledeč diagram nam predstavi TCP zaglavje z opisom posameznih polj v primarne namene ter tudi z opisom uporabe v namene razvitega muholovca:



Slika 2. TCP zaglavje

Izvorna vrata (source port: 16 bitov)
Številka izvornih vrat.

Ponorna vrata (destination port: 16 bitov)
Številka ponornih vrat.

Sekvenčna številka (sequence number: 32 bitov)
Sekvenčna številka prvega okteta podatkov danega segmenta (samo če je SYN zastavica nič). V primeru, ko imamo nastavljeno SYN zastavico, je to začetna sekvenčna številka (Initial Sequence Number - ISN), uporabljena za vzpostavljanje TCP povezave. Številka prvega okteta podakov je v tem primeru ISN+1.

Potrditvena številka (acknowledgment number: 32 bitov)
V primeru, ko imamo nastavljeno ACK zastavico, nam potrditvena številka predstavlja vrednost naslednje sekvenčne številke, katero pričakuje pošiljatelj med vzpostavljanjem TCP povezave. Ko je ta enkrat uspešno vzpostavljena, je potrditvena številka vedno prisotna.

Kazalec na podatke (data offset: 4 bitov)
Številčna vrednosti, katera nam pove na katerem odmiku so nahajajo vsebovani podatki danega TCP paketa.

Rezervirano (reserved: 6 bitov)
Rezervirano za potencialno uporabo v prihodnje. Mora biti nastavljeno na nič.

Kontrolni biti (control bits: 6 bitov) – od leve proti desni:

URG: Urgentna zastavica
ACK: Potrditev
PSH: Push fuknkcija
RST: Zahteva po resetiranju povezave
SYN: Zahteva po sinhronizaciji sekvenčnih števil
FIN: Označuje konec oddajanja podatkov

Okno (window: 16 bitov)
Število podatkovnih paketov, ki jih je prejemnik zmožen v danem trenutku prejeti.

Kotrolna vsota (checksum: 16 bitov)
Kontrolna vsota TCP paketa. Matematično gledano je to 16-bitni eniški komplement vsote celotnega TCP paketa. Če paket vsebuje liho število oktetov, se zadnjemu doda en oktet ničel z desne, tako da imamo za računanje vsote na voljo 16-bitno vrednost. Vendar je ta oktet ničel dodan samo v ta namen in se seveda ne pošilja preko mreže. Pred računanjem mora biti kontrolna vsota nastavljena na 0.

Urgentni kazalec (urgent pointer: 16 bitov)
Je številska vrednost odmika (od sekvenčne številke naprej) na zaporedje urgentnih podatkov danega paketa. Urgentni kazalec se upošteva samo za pakete, ki imajo nastavljeno URG zastavico.

Identifikacija paketa:
Vsak mrežni paket, ki vsebuje simx ukaz strežnika gonilniku ima polje TCP zaglavja `source port` nastavljeno na fiksno vrednost `SX_REQ_TCP_PORT`.

```
#define SX_TCP_PORT       (0x50)
#define SX_REQ_TCP_PORT   SX_TCP_PORT
```

Vsak mrežni paket, ki vsebuje odgovor (podatki zajete aktivnosti) simx gonilnika strežniku ima polje TCP zaglavja `source port` nastavljeno na fiksno vrednost `SX_REPLY_TCP_PORT`.

```
#define SX_REPLY_TCP_PORT   (0x1111)
```

1.1.5.2 UDP (User Datagram Protocol)

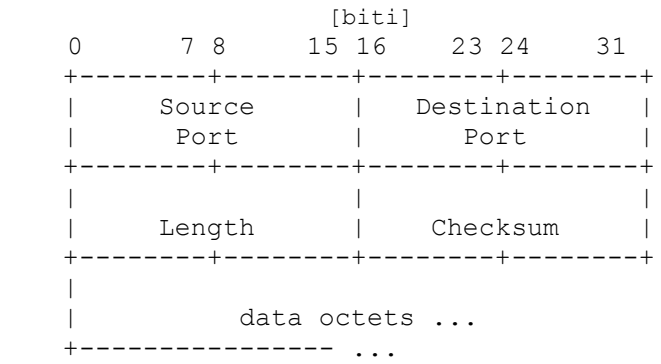
User Datagram Protocol (UDP) je še eden izmed ključnih gradnikov, ki sestavljajo internetni mrežni sklad. UDP omogoča mrežnim aplikacijam pošiljanje podatkovnih paketov, imenovanih tudi datagrami, po IP omrežju brez vnaprejšnje vzpostavitve komunikacijskega kanala ali podatkovne poti. Drugo poimenovanje za UDP je tudi Universal Datagram Protocol. Avtor protokola je David P. Reed, ki ga je leta 1980 formalno definiral v dokumentu RFC 786 [10].

Ker UDP uporablja preprost model prenosa brez implicitnega rokovanja ali vzpostavljana povezave, katera naj bi zagotavljala zanesljivost, pravilno zaporedje, je prenos preko UDP-ja relativno nezanesljiv, kjer so lahko podatkovni paketi sprejeti v nepravilnem zaporedju, manjkajoči ali celo podvojeni brez posebnega opozorila. UDP namreč predpostavlja, da je preverjanje in odpravljanje napak med prenosom bodisi nepotrebno ali izvajano na višjem, aplikacijskem nivoju ter se s tem izogne dodatnemu delu na transportnem nivoju mrežnega sklada. Kot tak se pogosto uporablja za komunikacijo aplikacijskih protokolov, kateri uporabljajo časovno kritične operacije z zahtevo po odzivnosti v realnem času. Namreč za tovrstne operacije je sprejemljivejši model delovanja sprejem novega paketa kot pa čakanje zakasnjenega. Za tiste aplikacijske nivoje, ki zahtevajo striktnejše preverjanje in odpravljanje napak med prenosom na transportnem nivoju, je na voljo Transport Control Protocol (TCP) ali Stream Control Transmition Protocol (SCTP), kateri so zasnovani ravno v te namene.

Dejstvo, da UDP ne potrebuje vzpostavljene transportne poti med odjemalcem in strežnikom, je zelo primerno za uporabo v primerih, ko mora strežnik odgovoriti velikemu številu odjemalcev. Za razliko od TCP nam UDP omogoča pošiljanje enega paketa na več različnih naslovov.

Nekaj primerov aplikacij, katere uporabljajo protokol UDP v transportne namene: Domain Name System (DNS), aplikacije za prenos multimedijske (video in audio) vsebine (IPTV,VoIP), Trivial File Transfer Protocol (TFTP) ter mnogo mrežnih igric.

Sledeč diagram nam predstavi UDP zaglavje z opisom posameznih polj v primarne namene ter tudi z opisom uporabe v namene razvitega muholovca:



Slika 3. UDP zaglavje

Izvorna vrata (source port: 16 bitov)
Številka izvornih vrat. V primeru da niso uporabljena mora imeti to polje vrednost nič.

Izvorna vrata (destination port: 16 bitov)
Številka ponornih vrat.

Dolžina (length: 16 bitov)
16-bitna vrednost, ki vsebuje dolžino celotnega paketa, tako zaglavja kot podatkov v bajtih. Najmanjša vrednost je 8 bytov, kolikor je veliko UDP zaglavje. Teoretična največja vrednost je 65,353 bytov, kar nam nanese

podatkovno kapaciteto UDP paketa 65,527 bytov, vendar je ta vrednost v praksi manjša, saj smo v primeru uporabe IP transporta omejeni še z velikostjo IP paketa, kjer na zaglavje odpade še dodatnih 20 bytov.

Kontrolna vsota (Checksum: 16bitov)
Kontrolna vsota UDP paketa, ki se uporablja za preverjanje napak tako za zaglavje kot vsebino paketa med mrežnim prenosom.

Identifikacija paketa:

Vsak mrežni paket, ki vsebuje simx ukaz strežnika gonilniku ima polje UDP zaglavja `source port` nastavljeno na fiksno vrednost `SX_REQ_TCP_PORT`.

```
#define SX_UDP_PORT       (0x1234)
#define SX_REQ_UDP_PORT   SX_UDP_PORT
```

Vsak mrežni paket, ki vsebuje odgovor (podatki zajete aktivnosti) simx gonilnika strežniku ima polje UDP zaglavja `source port` nastavljeno na fiksno vrednost `SX_REPLY_TCP_PORT`.

```
#define SX_REP_UDP_PORT   (0x1111)
```

1.1.6 Mrežna plast (IP v4,v6, ICMP)

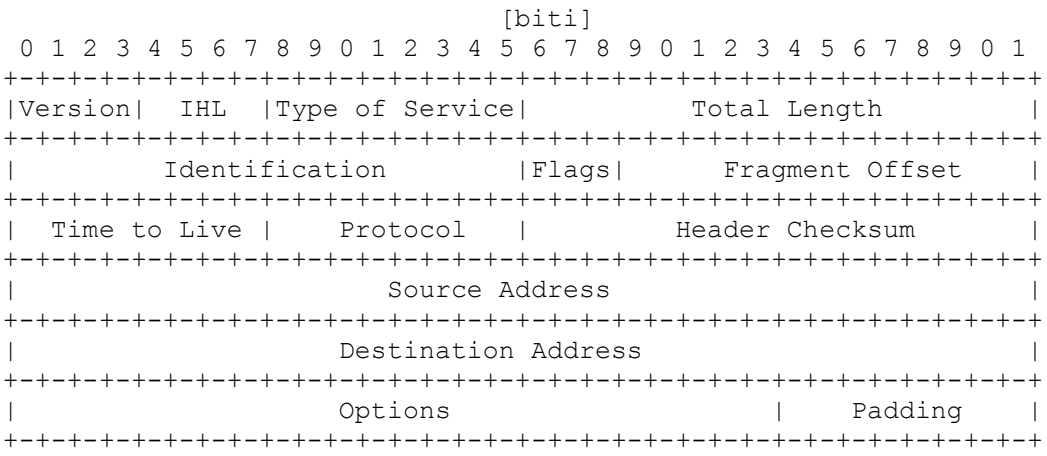
IP je primarni protokolni gradnik, ki sestavlja internetni mrežni sklad. Zadolžen je za usmerjanje in dostavo (različnih - glede na transportne protokole na spodnje ležečih plasteh) mrežnih paketov od izvornega do ciljnega računalnika samo na osnovi IP naslova, vsebovanega v zaglavju IP paketa. IP je zasnovan tako, da podpira več naslovnih struktur in modelov za gnezdenje različnih spodnje ležečih paketnih vsebin. Prva širše sprejeta in uporabljena verzija, imenova tudi Internet Protocol Version 4 (IPv4), je dan danes še vedno najpogosteje uporabljena različica na internetu, čeprav njena nadgrajena različica, poimenovana Internet Protocol Version 6 (IPv6) vztrajno pridobiva vedno večji delež aktivne uporabe na internetu.

1.1.6.1 IPv4

Internet Protocol version 4 (IPv4) je četrta verzija Internet Protocol-a in prva širše uporabljena in sprejeta za širšo uporabo. Skupaj z naslednjo različico Internet Protocol 6 sestavljata ključna protokolna gradnika interneta in sta kot taka daleč najbolj uporabljena protokola mrežne plasti internetnega mrežnega sklada.

IPv4 je podatkovno orientiran protokol, namenjen za uporabo v paketnem omrežju (npr. Ethernet). Zasnovan je tako, da zagotavlja dostavo paketa po najbolj optimalni poti, vendar ne nudi nobene garancije za dostavo paketa na ciljno točko, prav tako ne nudi nobene garancije za prejetje paketov v pravilnem zaporedju ali odpravljanje podvajanja le teh. Za tiste aplikacijske plasti, ki zahtevajo striktnejše preverjanje in odpravljanje napak med prenosom na transportnem nivoju, je na voljo Transport Control Protocol (TCP) ali Stream Control Transmition Protocol (SCTP), kateri so zasnovani ravno v te namene. Je pa potrebno izpostaviti, da nam IP nudi preverjanje integritete celotnega IP paketa s pomočjo kontrolne vsote, izračunane iz zaglavja.

Sledeč diagram nam predstavi IPv4 zaglavje z opisom posameznih polj v primarne namene ter tudi z opisom uporabe v namene razvitega muholovca:



Slika 4. IPv4 zaglavje

Verzija (version: 4bitov)
Prva polovica prvega polja zaglavja IP paketa je 4-bitna vrednost, namenjena informaciji o verziji IP paketa.

Dolžina IP zaglavja (Internet Header Length - IHL: 4bitov)
Druga polovica prvega polja IP zaglavja je njegova dolžina, ki vsebuje število 32-bitnih besed, iz katerih je le to sestavljeno. Ker lahko IPv4 zaglavje vsebuje spremenljivo število opcij, nam dolžina zaglavja služi tudi kot odmik do podatkov IP paketa. Najmanjša dolžina zaglavja je 5 besed, medtem ko je največja dolžina glede na to, da je sestavljeno iz 4 bitov, 15 besed.

Spremeljive storitve (Differentiated Services – DS: 8 bitov)
V preteklosti imenovano tudi tip storitve (Type Of Service - TOS). Po specifikaciji RFC 2476 [18] je bilo ponovno določeno kot polje z imenom spremenljive storitve ali pa tudi kot Explicit Congestion Notification (ECN) po najnovejši različici sprememb za IPv6 protokola definirano v specifikaciji RFC 3168 [19]. Namenjen je uporabi iz strani novejših internetnih tehnologij, ki zahtevajo propustnost in odzivnost v realnem času, kot je naprimer IP telefonija, prenosi video vsebine …
Začetni namen tega polja, še s starim imenom tip storitve, je bil oddajniku nastaviti zaželjen način prenosa mrežnih paketov na poti čez internet od oddajnika do naslovnika. Na primer, neki oddajnik lahko nastavi vrednost tega polja na preferiranje nizkih zakasnitev, medtem ko ima drugi tip storitve na izbiro nastavitev biti pripravljen na večjo zakasnitev, vendar ob večji zanesljivosti prenosa. Res je, da v praksi to polje nikoli ni bilo veliko uporabljeno, vendar je pa res tudi dejstvo, da je bilo narejeno veliko eksperimentalnega in raziskovalnega dela na temo, kako čim bolj smotrno uporabiti teh 8 bitov. Rezultate si lahko pogledamo v specifikaciji RFC 971 [18], kjer je med ostalimi podrobnostmi IP protokola podrobneje razdelana tudi vloga tega polja.

Celotna dolžina (Total Length: 16bitov)

Polje celotna dolžina vsebuje dolžino celotnega IP paketa, zajemajoč dolžino zaglavja in podatkov v bajtih. Najmanjša dolžina paketa je 20, kolikor znaša najmanjše možno zaglavje, največja vrednost tega polja je pa 65,535 bajtov. Najmanjša dolžina IP paketa, katerega naj bi znala obdelati (sprejeti in oddati) katerakoli mrežna naprava, je 576 bytov. Je pa potrebno upoštevati dejstvo, da lahko nekatera podomrežja med prenosom mrežnega paketa čezenj, predpisovati še dodatne omejitve (npr. najmanjša velikost Ethernet okvirja), ki zahtevajo fragmentacijo IP paketa. Namreč na nižjih plasteh (povezavni plasti) imamo lahko različne omrežne tehnologije za izvedbo povezave, od katerih je odvisno, kako dolga je maksimalna enota MTU (Maxsimum transmission unit). IP datagram se lahko na poti razbije na več manjših datagramov (fragmentacija). Sestavljanje majhnih datagramov (defragmentacija) poteka na ciljni strani. Po sestavljanju se datagram preda transportni plasti..

Identifikacija (identification:16 bitov)
Identifikacija polj služi za unikatno razlikovanje posameznih IP paketov.

Zastavice (Flags: 3 biti)
Sledijo 3 biti, kjer vsak predstavlja po eno izmed spodaj naštetih zastavic, uporabljenih za kontrolo ali identifikacijo paketa:

- Rezervirano; mora biti nič
- Ne fragmentiraj (DF)
- Več fragmentacije (MF)

V primeru da je nastavljena zastavica za prepoved fragmentacije (DF) in se nekje na poti paketa na danem usmerjevalniku pojavi zahteva po fragmentaciji, bo tak paket zavržen iz strani usmerjevalnika. V primeru fragmentacije morajo vsi paketi dane seje imeti nastavljeno zastavico MF z izjemo zadnjega.

Fragmentacijski odmik (Fragmentation offset: 13bitov)
Fragmentacijski odmik v enoti 8-bajtnih blokov pove odmik posameznega fragmenta glede na začetek orginalnega nefragmentiranega IP paketa. Prvi fragmentiran paket ima to vrednost seveda nič, medtem ko je največja vrednost odmika 65,528, kar je več kot dovolj, saj presega največjo dolžino IP paketa za 8 bajtov.

Dovoljeno število skokov (Time To Live – TTL: 8bitov)
8-bitna vrednost števila dovoljenih skokov varuje IP pakete pred neskončnim potovanjem po internetu v primeru usmerjanja v krogu. Ali drugače povedano, to število nam pove življenjsko dobo paketa. Vrednost se pri vsakem prehodu paketa čez usmerjevalnik zmanjša za ena in ko pade na nič, se paket zavrže. Oddajniku se v tem primeru pošlje ustrezno sporočilo ICMP. Omenimo še, da uporablja ravnokar opisano funkcionalnost znano diagnostično orodje `traceroute`.

Protokol (Protocol)
Polje protokol vsebuje informacijo o tipu uporabljenega (in gnezdenega) transportnega protokola v podatkovnem delu IP paketa. Vrednosti za vse danes

uporabljene protokole so definirane in predpisane v specifikaciji RFC 790 [21].

Kontrolna vsota zaglavja (Header Checksum: 16bitov)
Kontrolna vsota zaglavja je 16-bitna vrednost, ki se uporablja za preverjanje napak IP zaglavja, do katerih lahko pride med prenosom paketa po omrežju. Preverjanje se izvaja med vsakim preskokom na danem usmerjevalniku. V primeru napake zaglavja se tak paket takoj zavrže. Naj poudarim še dejstvo, da gre tu samo za preverjanje napake na nivoju zaglavja. Preverjanje integritete gnezdene vsebine je v domeni transportnega protokola, kateri imajo v te namene svoje različice polja za kontrolno vsoto.
Ker se pa vrednost TTL polja pri vsakem prenosu čez usmerjevalnik zmanjša, je potrebno kontrolno vsoto na vsakem usmerjevalniku tudi na novo izračunati.

Izvorni naslov (Source address)
Izvorni naslov protokola IPv4 je sestavljen iz štirih oktetov kateri nam dajo skupaj 32bitni naslovni prostor. Tukaj je potrebo opozoriti, da ni nujno da imamo opravka z dejanskim izvornim naslovom IP paketa, kajti če je bil paket generiran v lokalnem omrežju, ki uporablja NAT mehanizem na svojem robnem usmerjevalniku bomo v tem polju dobili naslov usmerjevalnika.

Končni naslov (Destination Address)
IPv4 naslov ponorne točke.

Možnosti (options: 32bitov)
Dodatno opcijsko zaglavje, katero (če je uporabljeno), sledi polju končnega naslova, vendar v praksi to ni ravno pogosto.

Identifikacija paketa:
Vsak mrežni paket, ki vsebuje simx ukaz strežnika gonilniku ima polje IP zaglavja `identification` nastavljeno na fiksno vrednost `SX_IP_REQ_ID`.

```
#define SX_IP_REQ_ID          (0x1234)
```

Vsak mrežni paket, ki vsebuje odgovor (podatki zajete aktivnosti) simx gonilnika strežniku ima polje IP zaglavja `identification` nastavljeno na fiksno vrednost `SX_IP_REPLY_ID`.

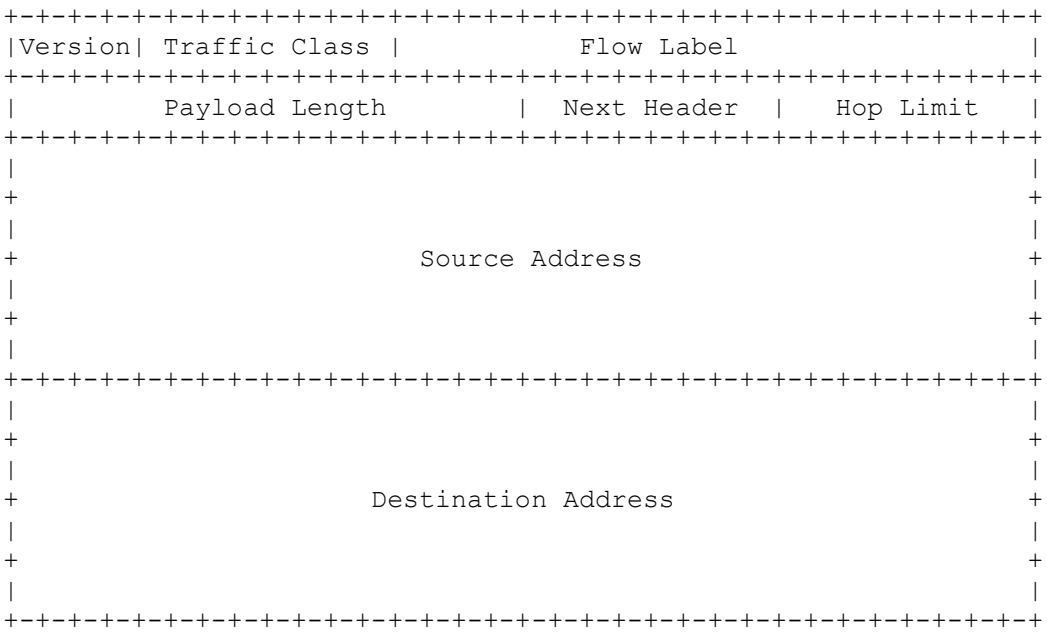
```
#define SX_IP_REPLY_ID        (0x4321)
```

1.1.6.2 IPv6

Internet Protocol version 6 je najnovejša različica internet protokola zasnovana kot naslednica IPv4. Seznam ključnih sprememb:

- Razširjeno naslovno polje
- Poenostavljena struktura zaglavja
- Izboljšana podpora za različne opcijske parametre in razširitve
- Podpora označevanju pretoka
- Avtentikacijska podpora

Sledeč diagram nam predstavi IPv6 zaglavje z opisom posameznih polj v primarne namene ter tudi z opisom uporabe v namene razvitega muholovca:



Slika 5. IPv6 zaglavje

Verzija (version: 4 bitov
Številka IPv6 verzije.

Prioritetni razred (traffic class: 8 bitov)
Prioritetna vrednost za dostavo IP paketa.

Oznaka za nadzor pretoka (flow label: 20 bitov)
Definira posebna navodila usmerjevalnikom na poti paketa od izvora do ponora.

Podatkovna dolžina paketa (payload length: 16 bitov)
Definira velikost podatkov, gnezdenih v paketu. Lahko je tudi 0 kar pomeni da je velikost paketa variabilna.

Tip transportnega protokola (next header: 8 bitov)
Definira, katerega transportnega protokolnega tipa je gnezden paket. Vrednosti so enake kot za IPv4.

Število dovoljenih prehodov (hop limit: 8 bitov)
8-bitna vrednost števila dovoljenih skokov varuje IP pakete pred neskončnim potovanjem po internetu v primeru usmerjanja v krogu. Ali drugače povedano, to število nam predpiše življenjsko dobo paketa. Namreč vrednost se pri vsakem prehodu paketa čez usmerjevalnik zmanjša za ena in ko pade na nič, se paket zavrže. Oddajniku se v tem primeru pošlje ustrezno ICMP sporočilo. Funkcionalno gledano je to polje ekvivalentno polju TTL IPv4 verzije protokola.

Izvorni naslov (source address: 32 bajtov)
The IPv6 address of the sending node.

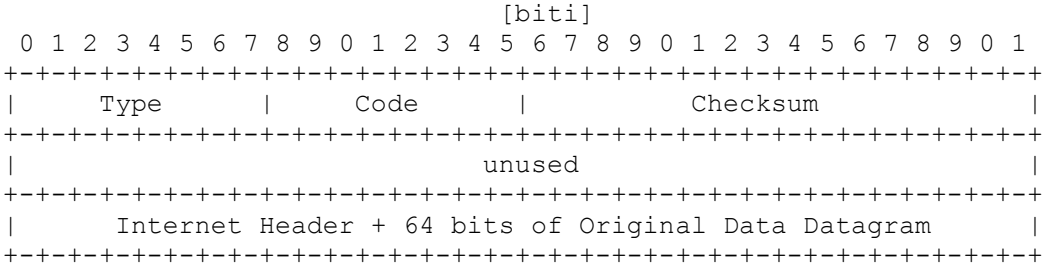
Končni naslov (destination address: 32 bajtov)
IPv6 verzija naslova ponorne točke.

1.1.6.3 ICMP (Internnet Control Message Protocol)

Internet Control Message Protocol (ICMP), definiran v dokumentu RFC 792 [9], je tesno povezan z mrežnim protokolom IP. Dejstvo je, da ICMP po definiciji spada v mrežno plast TCP/IP modela, vendar sem ga pri razvoju muholovca, z namenom nuditi čim večjo fleksibilnost, uporabil tudi v transportne namene.

ICMP sporočila, katera so gnezdena v IP paketu, so uporabljena za širok spekter analize in odpravljanja problemov, ki se pojavljajo med uporabo omrežja. Je pa tudi dejstvo, da če že obstajajo problemi med prenosom po danem omrežju, ne obstaja nobena garancija, da bo ICMP sporočilo dejansko prispelo na cilj. Nekaj poglavitnih nalog ICMP-ja:

- **Sporočanje napak na mreži,** kot so naprimer nedosegljivost posameznega računalnika ali celotnega dela mreže zaradi poljubnega razloga. Konkretno povedano TCP ali UDP paketi naslovljeni na vrata, za katerimi ni nobenega servisa, dobijo temu primeren ICMP odgovor.
- **Sporočanje nezadstne usklajenosti.** V primeru, da se na usmerjevalniku zaradi nezadovoljive prepustnosti začno kopičiti mrežni paketi, ima tak usmerjevalnik na voljo ustrezno ICMP sporočilo za obveščanje vseh oddajnikov o nastali situaciji. Predpisan odgovor oddajnika, po prejetju omenjenega sporočila, je zmanjšanje intenzivnosti oddajanja in s tem bolj usklajeno oddajanje in usmerjanje mrežnega prometa.
- **Pomoč pri iskanju napak.** ICMP nudi poizvedovalno sporočilo, za katerega je predviden odgovor. Zelo razširjeno diagnostično orodje ping uporablja funkcionalnost odgovarjanja na ICMP zahtevke za merjenje odzivnosti danega računalniškega sistema.
- **Pomoč pri diagnozi dosegljivosti .** V primeru, da pade vrednost TTL polja IP paketa na nič, se tak paket zavrže. Usmerjevalnik v tem primeru oddajniku pošlje ustrezno ICMP spročilo. Znano diagnostično orodje tracroute uporablja opisano funkcionalnost v namene mapiranja računalniškega omrežja.



Slika 6. ICMP zaglavje

Tip (Type: 8 bitov)
Vsebuje informacijo o tipu ICMP sporočila.

Koda (code: 8 bitov)
Vsebuje informacijo o nadaljnji klasifikaciji prejetega ICMP sporočila.

Kontrolna vsota ICMP zaglavja (ICMP header checksum: 16 bits)
Kontrolna vsota ICMP paketa. Matematično gledano je to 16-bitni eniški komplement vsote celotnega ICMP sporočila, začenši s tipom sporočila. Pred računanjem mora biti kontrolna vsota nastavljena na 0.

Vsebina (Data: spremenljiva dolžina)
Vsebuje podatke, določene s tipom in kodo sporočila.

Identifikacija paketa:

Vsak ICMP mrežni paket, ki vsebuje simx ukaz strežnika gonilniku vsebuje podpis v vsebini paketa:

```
#define SX_SIGNATURE (0xDECAFBAD) // signature value of simx packet
```

Prav tako mora vsak mrežni paket tipa ICMP, ki vsebuje odgovor (podatki zajete aktivnosti) simx gonilnika strežniku, vsebovati v zaglavju vsebine ravnokar omenjeni podpis.

1.1.7 Povezovalna plast (WLAN,Ethernet,Token Ring)

Povezovalna plast zagotavlja tako funkcionalne kot proceduralne gradnike, potrebne za prenos podatkovnih paketov med različnimi mrežnimi entitetami ter preverjanje in odpravljanje potencialnih napak, ki se lahko zgodijo na spodnjem fizičnem nivoju. V začetni fazi je bil ta nivo razvit za komunikacijo tipa točka-točka in točka-več točk, kar je značilno za telefonska omrežja. Koncept lokalnega omrežja, ki vključuje zahteve po dodatni funkcionalnosti tipa naslavljanja več točk hrati s simultanim dostopom, je bil razvit kasneje v okviru projekta IEEE 802 [11].

Tako WAN kot LAN omrežja gnezdijo prejete bite iz fizičnega nivoja v logična zaporedja, imenovana okvirji. Tukaj je potrebno izpostaviti, da ni nujno, da so vsi prejeti biti vključeni v zajete okvirje prenosnega nivoja, namreč nekateri biti so tam izključno za funkcionalne potrebe fizične plasti (npr. vsak peti bit FDDI bitnega niza).

1.1.8 Fizična plast

Fizična plast definira električne in fizične lastnosti samih mrežnih naprav oziroma definira, na kakšen način mrežne naprave uporabljajo fizični medij. Vključuje podrobnosti, kot so naprimer definicije pinov, napetostnih/tokovnih pragov, specifikacij mrežnih kablov, usmerjevalnikov, razdelilnikov, mrežnih kartic …

Za boljše razumevanje funkcionalnosti fizičnega nivoja v primerjavi s funkcionalnostjo prenosne plasti, lahko predpostavimo, da je primarna naloga fizičnega nivoja interakcija mrežne naprave s prenosnim medijem, medtem ko je naloga prenosne plasti interakcija večih (vsaj dveh) mrežnih naprav preko prenosnega medija. Torej, fizični nivo definira, kako mrežna naprava na eni strani odda podatke na prenosni medij, medtem ko na drugi strani predpisuje, kako naj naprava prebere prejete podatke iz prenosnega medija.

Glavne naloge fizičnega nivoja so:

- Vzpostavitev in podiranje povezave preko prenosnega medija
- Zagotavljanje procesov za porazdeljevanje prenosnih virov med večimi odjemalci prenosnega medija. Naprimer: tekmovanje za prenosni vir, kontrola prenosa ...
- Modulacija ali pretvorba med predstavitvijo podatka v digitalni obliki v sami napravi in temu podatku ustrezni vrednosti signala, kateri se dejansko pošilja po prenosnem mediju ali komunikacijskem kanalu. Kot primer tovrstnih signalov navedimo: radijski prenos, prenos preko optičnih vlaken, telefonskega omrežja ...

2 DEFINICIJA PROBLEMA: VDORI V RAČUNALNIŠKE SISTEME

V tem poglavju je predstavljen problem, kateri je predmet same naloge - vdori v računalniške sisteme ter tehnike obrambe pred njimi.

2.1 RAZVRŠČANJE NAPADOV

Poznamo veliko tipov mrežnih napadov, katerih klasifikacija po kategorijah je predstavljena v nadaljevanju.

2.1.1 Naključni napadi

Velika večina napadov na internetu je izvedenih s pomočjo avtomatiziranih orodij, najpogosteje uporabljenih s strani tehnično nepodkovanih vdiralcev, v žargonu imenovanih tudi kot script kiddie, kateri iščejo splošno znane ranljivosti javno dosegljivih mrežnih servisov. Karikirano povzeta primerjava bi bil primer vlomilca v hišo ali avto, kjer se le ta sprehaja od tarče do tarče ter išče nezaščitena (odklenjena) vrata, namreč z dovolj obsežnim poizkušanjem bo slej ko prej našel ranljivo tarčo.

Večina tovrstnih napadov se začne s skeniranjem celotnih blokov IP naslovov v poizkusu najti ranljiv računalniški sistem. Najbrž lahko z veliko verjetnostjo trdimo, da spada velika večina zajete aktivnosti, predstavljene v drugem delu praktičnega dela naloge, kjer so predstavljene dnevniške datoteke zajetih poizkusov napada prav v kategorijo naključnih napadov.

2.1.2 Neposredni napadi

Za direktni napad štejemo poizkus vdora s strani ponavadi tehnično podkovanega in izkušenega napadalca v vnaprej izbrano tarčo, kot je na primer spletna stran elektronske prodajalne, katera vsebuje veliko število podatkov o imetnikih kreditnih kartic. V primeru tovrstnega napada je ponavadi napadeno zelo omejeno število računalniških sistemov, na katerih se za vdor izkoristi še neznano ranljivost javno dostopnega servisa, ki teče na danem sistemu. Kot nazoren primer direktnega napada navedimo krajo 40 milijonov kreditinih kartic, ki se je zgodila spletni strani MasterCard International, kjer so sredi leta 2005 izvedeli, da je bilo v preteklem obdobju nekaj mesecev več kot 40 milijonov njihovih kreditnih kartic neavtorizirano uporabljenih [12].

Direktnih napadov se ne da preprosto zaznati s pomočjo konvencialnih orodji, kateri za analizo mrežnega prometa opozarjajo nadzornike omrežja na poizkuse vdorov. Kriteriji za pozitiven alarm napada temeljijo na podpisih že znanih tehnik napada, katere so, kot smo že prej omenili, redko uporabljene za direktne napade na skrbno izbrane tarče. Medtem ko lahko muholovec, nastavljen na računalniškem sistemu, z vabljivo vsebino zajame ravno te vnaprej neznane tehnike vodra v tak sistem, kjer lahko uporabimo rezultate analize zajetega materiala za izboljšavo zaščite ostalih produkcijskih sistemov. Ravno v ta namen sem v drugem delu praktičnega dela naloge postavil sistem z razvitim muholovcem na eni kontrolni točki testnega poligona, postavljenega v kontekstu dipolomske naloge. Podrobnejši opis postavljenega okolja ter rezultati so predstavljeni v nadaljevanju.

2.1.3 Izkoriščanje ranjivih mrežnih servisov

Večina programske opreme, odjemalcu dosegljive kot mrežni servisi preko interneta, (vsaj) v svojih začetnih fazah ni bilo razvite z namenom biti napisane kar se da varno in je kot taka ranljiva na različne poizkuse izkoriščanja le te. Za primer navedimo BSD tipe oddaljenih servisov (rlogin, rexec, rsh, wu-ftpd itd). V tem primeru je najboljši primer zaščite izklop vseh ranljivih servisov ali zamenjava z novejšimi neranljivimi verzijami.

2.1.4 Zavrnitev storitve

Zavrnitev storitve je tip napada, ki ima za cilj odpoved servisne aktivnosti napadenega javno dostopnega servisa za ostale uporabnike. Tovrstni napadi so najpogosteje izvršeni na mrežnem nivoju s sestavo in pošiljanjem paketov, vsebujoč podatkovne nize, kateri, ko jih servis na strežniku poizkusi prejeti, povzročijo odpoved servisne aktivnosti. Poznamo tudi primere napadov na aplikacijskem nivoju, kjer vnaprej pripravljeni ukazi za ciljno aplikacijo povzročijo odpoved delovanja le te.

2.1.5 Potvarjanje mrežnih naslovov

Napad s pretvarjanjem izvornega mrežnega naslova je ponavadi izvršen s ciljem, da se napadalec predstavi kot nekdo drug. Tipično se napadalec pretvarja, da je overjen odjemalec s potvarjanjem napadenemu sistemu znanih IP naslovov. Kot primer tovrstnega napada navedimo dobro znano ranljivost BSD rlogin servisa [13], kjer se je lahko napadalec, z ugibanjem in napovedovanjem sekvenčnih številk TCP zaglavja, predstavil in ugrabil že vzpostavljeno in avtorizirano sejo poljubnega odjemaleca. Za zaščito pred napadi s potvarjanjem je potrebno preverjati avtentičnost mrežnih paketov ter prepovedati usmerjanje paketov z neveljavnimi izvornimi IP naslovi. Poleg tega je v veliko pomoč za zaščito tudi uporaba sistemske mrežne programske opreme, ki zagotavlja čim večjo naključnost na nivoju kontrolnih mehanizmov povezave, npr: naključno izbrana sekvenčna števila TCP zaglavja, dinamična izbira vrat itd

2.1.6 Prisluškovanje

Prisluškovanje sodi med najpreprostejše oblike mrežnih napadov. Namreč tukaj imamo računalniški sistem, ki je skonfiguriran tako, da zajame vso aktivnost vidno na danem segmentu mreže, nakar je možno, s posebej v te namene napisano programsko opremo, izluščiti kritične koščke zajetega prometa, kot so naprimer uporabniška imena in pripadajoča gesla ostalih uporabnikov na danem mrežnem segmentu. Računalniška omrežja s topologijo vodila so še posebej ranljiva na napade s prisluškovanjem. Zato se je za zaščito pred tovrstnimi napadi priporočljivo izogibati ranljivim topologijam računalniških omrežij. Prav tako nam konfiguracija striktnejših pravil na IP nivoju na robnem požarenm zidu zmanjša verjetnost neavtoriziranega dostopa in verjetnost denial of service napadov na mrežnem nivoju. Poleg tega nam uporaba šifriranja mrežnega prometa nudi še dodaten nivo zaščite pred prisluškovanjem.

2.2 TEORETIČNI OPIS NAPADA

Povzetek teoretičnega primera vdora lahko razdelimo na več korakov ali faz:

2.2.1 Poizvedovanje

Najpogosteje je prvi korak napada na dano računalniško omrežje poizvedovanje o tarči napada, z namenom pridobiti čim več podatkov napadenega omrežja, kar je pogostokrat poimenovano kar podpis omrežja. Za to obstaja veliko različnih tehnik, katerih cilj je dobiti sledeče podatke o omrežju:

- Uporabljeni IP naslovni bloki
- Registrirana imena posameznih računalniških sistemov
- Dosegljivi računalniški sistemi
- Javno dostopni servisi
- Različice operacijskih sistemov in servisov
- Stopnja zaščite javno dostopnih servisov
- Aplikacijska struktura strežnikov za požarnim zidom
- Morebitne kritične podrobnosti, ki so bile nehote razkrite s strani nadzornikov napadenega omrežja med udejstvovanjem v javno dostopnih diskusijah

Prvi korak poizvedovanja je pridobiti logične lokacije napadenega omrežja, kar je relativno preprosto glede na dejstvo, da so registrirana imena napadenega omrežja dosegljiva preko javno dostopnih servisov, ki uporabljajo storitev WHOIS za poizvedovanje o lastništvu registriranih domen ali IP naslovov, kateri omogoča neavtoriziranemu uporabniku pridobitev velikega obsega detajlov o željenem računalniškem omrežju: od kontaktnih naslovov, telefonskih številk do registriranih domenskih imen itd.

2.2.1.1 Registrirana imena posameznih računalniških sistemov

Naslednji korak napada je tipično poizvedovanje za registriranimi domenskimi imeni posameznih računalniških sistemov napadenega računalniškega omrežja. Dan danes vedno manj, a v preteklosti velikokrat uporabljena tehnika za pridobitev seznama vseh registriranih imen, je bil prenos DNS območja za dano domeno, kjer dobi napadalec, v primeru premalo restriktivno nastavljenega DNS strežnika, seznam vseh registriranih imen v domeni napadenega omrežja. Imena niso ravno ključnega pomena za uspešen napad, vendar ga pogosto znatno olajšajo. Za konkreten primer napisanega predpostavimo, da najde napadalec spletni strežnik, na katerem teče Microsoft-ov strežniški paket IIS, kjer je anonimni uporabniški račun ponavadi imenovan po sledečem načinu `IUSR_<ime računalnika>`. Sedaj, če predpostvimo še, da je na tem strežniku nastavljena zaklenitev posameznih uporabniških računov, je vse kar mora narediti napadalec za uspešen napad na funkcionalnost spletenega servisa, pošiljati veliko število zahtev po avtentikaciji anonimnega uporabnika z neveljavnim geslom. Če je v dovolj kratkem času bilo generirano zadostno število tovrstnih zahtevkov, se bo anonimni uporabniški račun zaklenil, nakar lahko ob nadaljevanju pošiljanja neveljavnih avtentikacijskih zahtevkov napadalec povzroči neodzivnost spletnega strežnika tudi za legalen promet.

2.2.2 Skeniranje

2.2.2.1 Dosegljivi računalnikški sistemi

Od seznama registriranih domenskih imen je za napadalca še bolj zanimiv seznam računalniških sistemov napadenega omrežja, kateri so dejansko javno dosegljivi na internetu. V tej fazi napada se napadalec osredotoči na iskanje ranljivejših tarč iz

seznama. Dobiti listo dosegljivih računalnikov je lahko zelo preprosto opravilo ob predpostavki, da napadeno omrežje ne blokira vhodnega ICMP prometa na svojih požarnih zidovih, namreč že s preprosto sistemsko komando `ping` lahko preverimo dosegljivost posameznega računalnika. Za iskanje celotnih blokov IP naslovov lahko napadalec uporabi posebej napisano skripto, katera avtomatizira preverjanje ICMP odgovorov za vhodne sezname IP naslovov. Torej, če ima napadeno omrežje vklopljeno podporo za ICMP protokol, mora napadalec samo počakati, da skripta zgenerira seznam dosegljivih računalniških sistemov v napadenem omrežju.

Glede na to, da blokiranje ICMP prometa napadalcu znatno otežuje iskanje dosegljivih računalnikov napadenege omrežja, namreč v nasprotnem primeru mora napadalec za to nalogo uporabiti orodja, katera slonijo na analizi odgovorov ostalih dovoljenih protokolov (TCP,UDP), bi bil logičen sklep, da je ICMP promet blokiran za veliko večino zaščitenih omrežij, vendar temu ni tako. Namreč nekateri nadzorniki omrežij so prepričani, da bi s tem onemogočili podporno funkcionalnost IP omrežja, imenovano ugotavljanje optimalne vrednosti za PMTU (Path Maximum Transmition Unit), katere naloga je ugotavljanje maksimalne vrednosti velikosti mrežnega paketa na poti čez dano podomrežje, za kar se uporablja ICMP sporočila. Vendar temu ni tako. Namreč z blokiranjem samo ICMP sporočil, katera so uporabljena za iskanje dosegljivih sistemov (Echo in Reply), nima nobenega vpliva na zgoraj opisano funkcionalnost iskanja optimalne PMTU vrednosti.

V primeru, da napadalec nima na voljo ICMP protokola, se je primoran opreti na orodja, ki analizirajo odgovore na različno sestavljene zahteve TCP paketov. Najpreprostejša metoda je seveda poizkus vzpostavitve polne TCP povezave na poizvedovana vrata. Za vsako uspešno vzpostavljeno povezavo je napadalcu jasno, da v ozadju teče javno dosegljiv računalniški sistem, katerega se da napasti. Ker pa je tak način preverjanja povezav zelo opazen in ponavadi logiran na večih nivojih, namreč vzpostavljena je polna TCP povezava, se napadalci tega načina ne poslužujejo prav pogosto. Obstaja še veliko drugih tehnik preverjanja dosegljivosti strežnikov in iskanja javno dostopnih servisov napadenega sistema, ki so veliko bolj prikrite in manj opazne, vendar je podroben opis vseh tehnik preobsežen za to poglavje. Naštejmo jih samo nekaj s kratkim opisom posamezne tehnike:

- SYN – Najpogosteje uporabljena tehnika preverjanja dosegljivosti računalnika in pripadajočih servisov je vzpostavljanje polodprtih TCP povezav, kjer napadalec s pošiljanjem TCP paketov z nastavljeno zastavico SYN preverja dobljene odgovore:
    - paket z nastavljenima zastavicama ACK in SYN pomeni odprta vrata,
    - paket z nastavljeno zastavico RST pomeni zaprta vrata,
    - brez odgovora pa ponavadi pomeni, da so vrata filtrirana, kar pomeni, da je bil ves promet na ta vrata namenoma ignoriran.
- ACK – preverjanje konfiguracije požarnega zidu. Tu za željena vrata dobimo odgovor ali je dostop filtriran ali ne, o dejstvu, ali je servis za preverjanimi vrati dostopen ali ne, tu ne izvemo nič.
- Window – podobno kot prej opisana tehnika ACK, le da tu dobimo tudi informacijo o stanju servisa, ki mogoče posluša za preverjanimi vrati. Vrednost drsečega okna vrinjenega RST paketa (samo na določenih mrežnih napravah in operacijskih sistemih), je za zaprta vrata 0, medtem ko je za

odprta vrata ta vrednost večja od nič, kar je zadosten kriterij za ugotavljanje stanja (odprta/zaprta) nefiltriranih vrat.

- Null – preverjanje statusa vrat s pošiljanjem TCP paketa, kateri ima vse zastavice nastavljene na vrednost nič. Po definiciji TCP protokola RFC 793 [14] naj bi odgovor na TCP zahtevo, ki nima nastavljene vsaj ene izmed sledečih zastavic: SYN,RST ali ACK, vedno vseboval zastavico RST v primeru, da so vrata zaprta ali odgovora na tako zahtevo sploh naj ne bo, če so vrata odprta. Ravnokar opisana luknja v specifikaciji TCP protokola je uporabljena tudi v naslednjih dveh opisanih tehnikah: FIN in Xmas.
- FIN - preverjanje statusa vrat s pošiljanjem TCP paketa, kateri ima nastavljeno samo vrednost zastavice FIN.
- Xmas - preverjanje statusa vrat s pošiljanjem TCP paketov, kateri imajo nastavljene samo vrednosti zastavic FIN,URG in PSH.
- Maimon – preverjanje statusa vrat s pošiljanjem FIN in ACK paketov ter analizo razlike v TTL vrednostih odgovora.

Poleg tega je napadalec pozoren tudi na povezave s statusom zavrnjeno, namreč poizkus povezave na vrata, za katerimi ne posluša noben servis, bo zavrnjen, kar spet daje dodaten vplogled napadalcu na konfiguracijo požarnega zidu. Poglejmo si teoretičen primer poizvedovanja dveh vrat, na katerega dobimo sledeč status:

```
Port 192.168.2.30:443 refused

Port 192.168.2.30:80 open
```

Tukaj je napadalcu jasno, da za vrati 80 po vsej verjetnosti posluša spletni servis, medtem ko je dostop do vrat 443 (https) dovoljen, vendar zavrnjen, ker za temi vrati ni delujoče storitve, kar pa drugače povedano pomeni, da ima požarni zid po nepotrebnem dovoljen dostop (luknjo) v svoji konfiguraciji.

2.2.2.2 Javno dostopni servisi

Skeniranje odprtih vrat na danem računalniškem sistemu (portscaning) je najpogostejša tehnika ugotavljanja javno dostopnih servisov, ki tečejo na tem sistemu. To omogoča napadalcu sestaviti seznam potencialnih vektorjev napada. Sledeča tabela prikazuje seznam pogosto preverjanih vrat v prvi fazi napada:

Vrata	**Servis**
20	FTP data channel
21	FTP control channel
23	Telnet
25	SMTP
53	Connection-oriented DNS
80	HTTP
88	Internet Key Exchange
110	POP3 (Mail)
135	Windows RPC Endpoint Mapper
137	NetBIOS Name Service

Vrata	Servis
139	NetBIOS session
389	LDAP
443	HTTP/S
445	Common Internet File System / native SMB on Windows 2000 and higher
636	LDAP over SSL
1433	SQL Server (1434 if your port scanner can do UDP)
1723	PPTP
3268	LDAP to a Windows Global Catalog Server
3389	Remote Desktop Protocol (Windows Terminal Services)

2.2.2.3 Različice operacijskega sistema in servisov

Zelo uporabna informacija za nadalnje načrtovanje uspešnega napada na izbrani računalniški sistem so verzije javno dostopnih servisov, ki tečejo na tem sistemu. Obstaja več načinov, kako to dobiti. V večini primerov nam bo sama aplikacija, ki teče za določenimi vrati, razkrila željeno. Naprimer sendmail servis nam ob ročni povezavi na vrata 25 v pozdravnem sporočilu izpiše poleg ostalih podatkov tudi svojo verzijo, nakar lahko napadalec preveri, če obstajajo kakšne znane ranljivosti za dotično verzijo. V primeru ranljive verzije ima napadalec vse potrebno za začetek dejanskega napada. Od tu naprej lahko izkoristi to ranljivost za pridobitev (začetnega) nedovoljenega dostopa v napaden računalniški sistem. V primeru, da servisi ne razkrivajo informacije o različici aplikacije, je možno s posredovanjem določenih zahtev aplikaciji in analize dobljenih odgovorov izluščiti približne zaključke o tipu in različici strežniške aplikacije. Ta tehnika je zelo primerna in uporabljena v avtomatiziranih orodjih za iskanje sistemskih ranljivosti in orodij za prepoznavanje tipa operacijskega sistema, ki teče na opazovanem računalniškem sistemu. Napadalec pa ima vedno možnost, da kar na slepo poizkuša izkoristiti predvidevano ranljivost ter opazuje rezultate. Ravnokar opisan pristop je prav tako uporabljen v avtomatiziranih orodjih za preverjanje sistema za ranljivosti odpovedi delovanja storitve. Namreč, če je storitev po napadu še vedno aktivna, je jasno, da opazovan sistem ni ranljiv na tovrstni napad.

Potrebno je tudi izpostaviti, da ravnokar opisane tehnike napada izkoriščajo servisne ranljivosti, ki že po definiciji niso bile predvidene za vsakodnevno uporabo. Posledično lahko povzročijo destabilizacijo ali celo sesutje napadenega servisa ali celotnega računalniškega sistema, kar je seveda zelo opazen dogodek, ki daje nadzorniku omrežja slutiti, da se dogaja nekaj sumljivega z nadzorovanim omrežjem. Napadalec se tega dobro zaveda, zato mu ponavadi služijo tovrstne tehnike kot zadnji poizkus vdora po tem, ko odpovejo vsi ostali načini pridobitve neavtoriziranega dostopa..

2.2.2.4 Aplikacijska struktura strežnikov za požarnim zidom

Pridobitev čim večjega vpogleda v aplikacijsko strukturo, ki je postavljena na internem omrežju za požarnim zidom, je iz napadalčeve perspektive zelo zaželjen podatek.

Za nazorno predstavitev opisanega si poglejmo izmišljen teoretičen primer, kjer predpostavimo, da uporablja spletni strežnik napadenega omrežja slabo napisano (ali

vzdrževano) spletno aplikacijo za serviranje spletnih aktivnosti napadene organizacije ali podjetja, ki je značilna po neki lastnosti, recimo načinu poimenovanja svojih datotek ali dizajnu spletne strani, tako da ima napadalec iz teh značilnosti možnost vedeti, za katero aplikacijo gre. Ker je predvidena aplikacija, kot smo prej že omenili, slabo napisana (ali vzdrževana), je napadalcu dana možnost, da izkoristi kakšno izmed ranljivosti, značilnih za to aplikacijo. Za nadaljevanje ponazoritve recimo, da ima opisana aplikacija problem s preverjanjem http zahtev po datotekah in da napadalec v brskalniku zahteva serviranje konfiguracijske datoteke system.settings, ki se nahaja na korenskem imeniku spletnega strežnika. Ker jo spletni strežnik zaradi validacijske napake spletne aplikacije servira napadalcu, dobi le ta vpogled v celotno vsebino sistemske konfiguracije internega omrežja, kjer je v najboljšem primeru opisana samo topologija omrežja z imeni računalnikov in bazami, ki tečejo v ozadju, v najslabšem primeru so pa lahko vsebovani še veliko bolj kritični podatki, kot naprimer uporabniška imena in gesla, uporabljena za dostop spletnega strežnika do podatkovnih baz.

2.2.2.5 Morebitni kritični detajli kateri so bili nehote razkriti iz strani nadzornikov napadenega omrežja med udejstvovanjem v javno dostopnih diskusijah

Zelo zanimivo je dejstvo, da lahko napadalec izve veliko potencialno kritičnih detajlov o omrežju, ki ga napada, iz javno dostopnih novičarskih skupin ali forumov, kjer lahko nadzornik napadenega omrežja pri reševanju vsakodnevnih problemov nehote razkrije kritične informacije o omrežnih napravah, konfiguracijah požarnih zidov itd. Znani so tudi dogodki, ko je podjetje, z namenom, da se pohvali, kako bodo nadgradili in s tem dodatno zavarovali svoje omrežje, to novico objavilo v računalniški reviji ali na spletu in s tem nehote dalo napadalcu vpogled v načrte nadgradnje in morebitne pomanjkljivosti le te [15].

2.2.3 Pridobitev neavtoriziranega dostopa

Naslednji korak napada je pridobitev neavtoriziranega dostopa do napadenega strežnika, preko katerega napadalec v nadaljevanju poizkuša pridobiti priviligirane (nadzorniške) pravice, ki mu omogočajo popolno kontrolo sistema. Za dosego cilja ima dve možnosti:

- Napad z izkoriščanjem ranljivosti javno dosegljive storitve. Glede na različice javno dostopnih servisov, zbrane v prejšnjih fazah napadov, lahko napadalec preverja ali so le te ranljive. V primeru nevzdrževanega sistema s starejšimi (in ranljivimi) različicami programske opreme se tovrstno ranljivost izkoristi za pridobitev neavtoriziranega dostopa (ponavadi ukazna vrstica) do sistema. Uporabniške pravice takega dostopa so odvisne od uporabniškega konteksta, v katerem napadena storitev teče. V najslabšem primeru je to nadzorniški kontekst, kar pomeni, da ima s takim dostopom napadalec že kontrolo nad sistemom in lahko preskoči naslednjo fazo napada, to je pridobivanje privilegiranih pravic in lahko kar nadaljuje s skrivanjem dokazov o vdoru in zagotavljanjem (skritega) dostopa v prihodnje.
- Napad s preverjanjem gesel. V primeru, da je napaden sistem dobro vzdrževan in da kot tak nima nameščenih ranljivih storitev, ima napadalec še vedno možnost napada s poizkušanjem avtorizacije z gesli iz vnaprej skrbno pripravljene množice najbolj verjetnih. Tukaj je potrebno izpostaviti dejstvo, da je to lahko zelo časovno zahteven korak, pri katerem je verjetnost uspešne avtorizacije pogojena z različnimi dejavniki, med katerimi je ključnega

pomena pogostost uporabe posameznega gesla iz pripravljene množice. Bolje, ko je pripravljena množica gesel (ob upoštevanju domenskih in jezikovnih značilnosti napadenega sistema), večja je verjetnost uspešnega napada. Seveda pa je za uspešen napad z gesli potreben tudi veljaven uporabniški račun, katerega mora napadalec dobiti v prej omenjenih fazah napada (poizvedovanje itd).

2.2.4 Izkoriščanje ranljivosti na sistemu za pridobitev privilegiranih pravic

Naslednji cilj vdiralca, po pridobitvi začetnega (neprivilegiranega) dostopa do sistema, je izkoriščanje lokalnih ranljivosti na sistemu, z namenom pridobitve nadzorniškega dostopa in s tem popolna kontrola nad sistemom. Praktično je postopek enak prejšnjemu koraku s to razliko, da napadalec tukaj preveri ranljivosti lokalnih različic, nameščenih na sistemu, nakar zopet v primeru slabo vzdrževanega sistema (če je napadalec prišel do te točke, je to zelo verjetno) izkoristi le te za pridobitev privilegiranih pravic.

2.2.5 Brisanje dokazov vdora

Ko pride napadalec do nadzorniškega dostopa na napadenem sistemu, ponavadi poskrbi, da ostane vsa neavtorizirana aktivnost (dnevniki vdora ali ostalih nepravilnosti), katera ga bi lahko izdala, kar se da skrita in ne beležena. V ta namen ima napadalec na voljo veliko orodij za manipulacijo različnih tipov (na različnih platformah) dnevniških datotek [28].

2.2.6 Zagotavljanje dostopa v prihodnje

Po uspešnem vdoru zagotovitve popolnega nadzora nad sistemom in brisanju dokazov o neavtorizirani dejavnosti ostane napadalcu še zagotovitev (prikritega) dostopa do tega sistema v prihodnje. Tudi za to nalogo obstaja na spletu širok spekter orodij [29], [30].

2.3 KLASIFIKACIJA KORAKOV PREPREČEVANJA NAPADOV

2.3.1 Zaznavanje napada

Zaznavanje mrežnih napadov je nečeloma podobno alarmnim sistemon, ki varujejo hiše, poslovne objekte, banke … Ko vlomilec vdre v varovan objekt, se sproži alarm in o tem obvesti nadzorne organe. V domeni računalniške (mrežne) varnosti imajo to vlogo orodja za odkrivanje mrežnih napadov, ki z analizo mrežnega prometa in analiziranjem aktivnosti v sistemskih dnevniških datotekah sprožajo alarme ob detekciji podpisa znanega napada.

Največji problem tovrstnih sistemov so lažni alarmi ali še huje nedetektirani poizkusi dejanskih napadov na varovan računalniški sistem. Obstaja velika verjetnost proženja alarma o sumljivi ali nevarni aktivnosti na podlagi legalnega produkcijskega prometa na opazovani mreži. Verjetnost lažnih alarmov je še toliko večja na omrežjih, skozi katera se pretakajo velike količine prometa. Visoka raven mrežnega prometa lahko vodi tudi do nezaznavanja dejanskega napada, ker sistem za detekcijo le teh ni sposoben takojšnje obdelave opazovanega prometa (zato se nekateri paketi spuščajo) in procesa analize vsebovanja znanih vzorcev napada.

2.3.2 Preprečevanje napada

Za preprečevanje mrežnih napadov se uporabljajo pazljivo, temeljito in zadosti restriktivno napisana pravila požarnih zidov, katera ščitijo notranje dele omrežja pred posegi od zunaj. Poleg tega je potrebna temeljita disciplina zagotavljanja najnovejših verzij javno dostopnih servisov.
Dodana vrednost uporabe muholovca v tej kategoriji obrambe je relativno majhna, saj nam muholovec v primeru naključnega napada ne pomaga prav dosti, saj so ti napadi večinoma avtomatizirani in poleg zaznave le tega ne pridobimo nič drugega. Deloma nam lahko muholovec doprinese k zaščiti omrežja v primeru direktnega napada, kjer lahko zamoti napadalca, da se ukvarja z nastavljenim muholovec sistemom, ki je brez realne vrednosti in s tem pridobimo na času za zaščito produkcijskega sistema. Vendar je pisana rešitev samo delna, kajti deluje samo v primeru, da se napadalec prvo spravi na nastavljen muholovec sistem in ne obratno.
V primeru, da nadzornik javno objavi uporabo muholovcev na varovanem omrežju, posledično obstaja tudi verjetnost zmanjšanja verjetnosti napada na to omrežje. Vendar je ta aspekt dodane vrednosti muholovca pri preprečevanju napada bolj kot predmet psihologije in kot tak preveč abstrakten za konkretno oceno njegovega doprinosa k računalniški varnosti.

2.3.3 Odgovor na napad

Po uspešnem odkritju napada se je potrebno naučiti čim več o napadu in dobljene informacije uporabiti za zaščito pred tovrstnimi napadi v prihodnje.

V tej kategoriji obrambe je dodana vrednost muholovca velika. Muholovec nam v primeru napada nudi konkreten vpogled napadalčeve aktivnosti, kar nam omogoča natančno analizo neavtorizirane aktivnosti, uporabljene tehnike vdora, motivov, zakaj je do napada sploh prišlo itd.

Za poenostavljanje analize zajete aktivnosti so računalniški sistemi z muholovcem ločeni od produkcijskih in je vsaka aktivnost na takem sistemu po definiciji neavtorizirana in kot taka predmet nadaljnje preiskave, katera nam lahko nudi nove, še učinkovitejše načine mrežne zaščite. Več teoretičnega in praktičnega ozadja o konceptu muholovec sledi v nadaljevanju naloge.

3 ORODJA ZA ODKRIVANJE NAPADOV

Preden nadaljujemo s teoretičnim pregledom teorije muholovcev, si za uvod poglejmo nekaj (konvencionalnih) orodij za detekcijo in preprečevanje mrežnih napadov.

3.1 PREDSTAVITEV

Orodje za odkrivanje napadov (Intrusion Detection System - IDS) je programska ali strojna oprema, katere cilj je detekcija neavtoriziranih poizkusov dostopa, manipulacije ali zanikanja servisne storitve opazovanega računalniškega sistema. Za tovrstnimi napadi so lahko izkušeni posamezniki, avtomatizirana aktivnost zlonamerne programske kode (malware) ali celo nezadovoljni zaposleni. Šifriran mrežni promet je velikanska omejitev orodij za detekcijo napada, saj je neposredna analiza mrežnega prometa nemogoča.

Orodja za detekcijo napadov lahko detektirajo več različnih tipov zlonamerne aktivnosti na opazovanem omrežju, kot so naprimer mrežni napadi na javno dostopne servise, aplikacijski napadi na servisne (večinoma spletne) storitve, lokalni napadi na nivoju operacijskega sistema, z namenom povišanja prioritete dostopa, nepravilne prijave in dostope do datotek ali podatkovnih baz z občutljivo vsebino. Tako orodje je ponavadi sestavljeno iz več komponent:

- senzor – za zajem dogodkov, kateri so predmet nadaljne analize
- konzola – za pregled in kontrolo nastavljenih senzorjev in generiranih alarmov
- analiza dogodkov – od senzorja sprejema zajete dogodke, jih vpisuje v podatkovno bazo, kjer se vsebina prejetih dogodkov preveri s podpisi znanih vrst napadov in se na podlagi le teh in po konfiguriranih pravilih generira ustrezne alarme

Obstaja več kategorizacij tovrstnih orodij glede na tipe in lokacije senzorjev, uporabljanje metodologije za generiranje alarmov ter fleksibilnosti pri nastavljanju pravil. Pri preprostejših različicah orodja imamo vse tri komponente velikokrat združene v eno napravo.

3.2 TIPI ORODIJ ZA ODKRIVANJE NAPADOV

Kot smo že omenili na koncu prejšnjega poglavja, poznamo več kategorizacij orodij za odkrivanje napadov, od katerih je najbolj pogosta mrežna varianta, kjer imamo senzorje postavljene na ključne točke opazovanega omrežja, najpogosteje v demilitarizirani coni omrežja (DMZ) ali na vhodnih točkah varovanega omrežja. Senzorji zajemajo ves mrežni promet in analizirajo vsak zajeti paket s podpisi znanih vzorcev napadov.

Poleg mrežne različice poznamo tudi orodja za odkrivanje napada, ki bazirajo na analizi prometa na podatkovnem in aplikacijskem nivoju (PIDS in APIDS), kjer se preverjajo in iščejo vzorci škodljive podatkovne vsebine ali neveljavni ukazi danega jezika (npr. SQL). Nazadnje pa omenimo še sisteme odkrivanja napadov na nivoju operacijskega sistema, kateri nadzorujejo vso lokalno aktivnost opazovanega računalniškega sistema, na katerem je nameščeno tako orodje. Seveda pa poznamo tudi hibridne variante zgoraj opisanih orodij.

3.2.1 Mrežni sistem odkrivanja napadov (Network based IDS)

Mrežni sistemi za detekcijo napadov so neodvisna platforma za identifikacijo znanih vzorcev napada z analiziranjem mrežnega prometa v opazovanem segmentu omrežja. Dostop do mrežnega prometa dobijo z direktno povezavo preko huba, usmerjevalnika ali kar preslikave vrat. Najbolj znani primer mrežnega sistema za detekcijo napadov, imenovanega tudi *de facto* standard za IDS orodja, je Snort [ii], katerega sem tudi uporabil v praktičnem delu naloge.

3.2.2 Protokolni sistem odkrivanje napadov (PIDS)

Protokolni sistemi za detekcijo napadov so tipično sestavljeni iz senzorja, ki pred opazovanim računalniškim sistemom zajema in analizira komunikacijo med odjemalcem in strežnikom. Ponazorjeno na primeru spletnega strežnika bi to pomenilo nadzor HTTPS toka podatkov ob koreliranju vhodnih HTTP zahtevkov odjemalca glede na ščiteno vsebino na strani spletnega strežnika. Seveda bi ob uporabi HTTPS tak sistem moral opazovati dekriptirano mrežno vsebino, se pravi, tik preden jo po odobritvi s strani PIDS prejme spletni strežnik za predstavitev ali serviranje spletne vsebine.

3.2.3 Aplikacijski sistemi za odkrivanje napadov (APIDS)

Aplikacijski sistemi za detekcijo napadov so tipično postavljeni med podatkovne strežnike, kjer analizirajo promet med aplikacijami. Na primer transakcijska komunikacija med spletnim strežnikom in podatkovno bazo v ozadju bi bila nadzorovana in analizirana za vzorci (npr. tipa SQL) neveljavnega ali škodljivega prometa.

3.2.4 Sistemi odkrivanja napadov na nivoju OS (HIDS)

Sistemi detekcije napadov na nivoju operacijskih sistemov so zasnovani tako, da identificirajo napade z analizo sistemskih klicev, aplikacijskih in sistemskih dnevnikov, analizo sprememb na datotečnem nivoju (za kritičnimi datotekami, kot so sistemske knjižnice, gesla, podatkovne baze itd). Primer tovrstnega orodja je OSSEC ter TripWire [iii]. Slednji je uporabljen v praktične namene naloge in je malo obširneje razložen v nadaljevanju.

3.2.5 Hibridni sistemi odkrivanja napadov (HIDS)

Hibridni sistemi za detekcijo napadov kombinirajo dve ali več zgoraj opisanih variant tovrstnih sistemov. Primer takega orodja je Prelude [iiii].

3.3 PASIVNI TER AKTIVNI SISTEMI ZA ODKRIVANJE NAPADOV

Pasivni sistemi za detekcijo napada s senzorjem zajemajo in analizirajo mrežni promet in v primeru zaznanega vzorca napada samo generirajo alarm, katerega zabeležijo in po potrebi pošljejo nadzorniku. Medtem ko aktivni sistemi za detekcijo napada, imenovani tudi sistemi za preprečitev napada (Intrusion Prevention System - IPS), odgovorijo na sumljivo ali škodljivo aktivnost s prekinitvijo ali avtomatsko rekonfiguracijo pravil na požarnem zidu za to povezavo. Ta aktivnost je lahko avtomatska ali pa čaka na odobritev nadzornika varovanega omrežja. Sistem, kateri

ob zaznanem napadu prekine povezavo, sodi tudi v kategorijo požarnih zidov aplikacijske plasti.

Čeprav spadajo sistemi za detekcijo napadov med skupino orodij za zaščito računalniških omrežij, kot na primer požarni zidovi, jih ne smemo enačiti z njimi. Požarni zid namreč varuje notranje, zaščitene dele omrežja s filtriranjem vhodnega in izhodnega prometa, z namenom čim bolj omejiti in preprečiti neavtorizirane posege v notranje predele varovanega omrežja, vendar nima nobenega mehanizma za zaznavo potencialnih napadov. Medtem ko sistemi za detekcijo le teh na osnovi analize mrežnega prometa generirajo alarme o zaznanih poizkusih napada, tako od zunaj (internet) kot od znotraj (intranet) ali celo lokalno na strežniku.

Za zaključek omenimo še obstoj hibridnih sistemov med aktivnimi in pasivnimi sistemi, kateri so zasnovani tako za detekcijo napada in tudi odgovor nanj, imenovanih tudi IDPS (Intrusion Detection and Prevention System - IDPS).

3.4 KRITERIJI ZAZNAVANJA NAPADA

Vsi sistemi za detekcijo napadov uporabljajo eno izmed dveh tehnik ali kriterijev za generiranje alarmov: statistično analizo ali iskanje znanih vzorcev (podpisov) napada.

3.4.1 Vzorci napadov

Sistemi za detekcijo napadov na osnovi vzorcev uporabljajo bazo znanih vzorcev napada, imenovanih tudi podpis napada. Večina mrežnih napadov ima značilen vzorec aktivnosti, kateri se da relativno preprosto opisati in shraniti v podatkovni bazi za namene zaznavanja le teh.

3.4.2 Anomalija (statistično gledano) mrežnega prometa

Poznamo tudi sisteme za detekcijo napadov, ki uporabljajo statistično analizo mrežnega prometa, na osnovi katere iščejo nenavadne vzorce aktivnosti, ki odstopajo od statističnih parametrov mrežnega prometa v normalnih okoliščinah. Vsako odstopanje od normale označijo kot anomalijo, na osnovi katere se generira alarm o sumljivi, potencialno škodljivi aktivnosti na opazovanem omrežju.

3.5 OMEJITVE ORODIJ ZA ODKRIVANJE NAPADOV

3.5.1 Splošne omejitve

Nekatere vrste napadov so relativno preproste za uspešno detekcijo, kamor večinoma sodijo zgoraj opisani naključni napadi, izvedeni s pomočjo avtomatiziranih orodij, najpogosteje uporabljenih s strani tehnično nepodkovanih vdiralcev, kateri iščejo splošno znane ranljivosti naključnih javno dosegljivih mrežnih servisov.

Vendar pa, splošno gledano, detekcija mrežnega napada predstavlja zahteven problem. Kot prvo omejitev navedimo, da je že leta 1987 Fred Cohen [16] dokazal, da je detekcijo računalniškega virusa (v smislu, da ločimo namene opazovanega programa po kriteriju ali so zlonamerni ali ne) praktično nemogoče učinkovito implementirati. Kot drugo fundamentalno omejitev pa omenimo dejstvo, da se nekaterih tipov napadov ne da odkriti (ali je zelo zahtevno ali praktično nemogoče).
Poleg že omenjenih omejitev se tu pojavi še problem definicije orodij za detekcijo napada. Večina sistemov škodljivo aktivnost samo beleži, nekateri sistemi so

konfigurirani tako, da blokirajo škodljivo povezavo, medtem ko ostali v ekstremnem primeru kar izključijo napaden sistem, z namenom preprečiti nadaljno škodo. Zadnje opisan primer izključitve napadenega sistema dopušča možnost napadalcem za napad zanikanja servisne storitve napadenega sistema, kar je samo po sebi že dovolj velika pomanjkljivost, poleg tega pa tako orodje prevzame tudi vlogo mehanizma za kontrolo dostopa, kar je že samo po sebi zelo kompleksno in dovolj zahtevno opravilo. Namreč mehanizmi kontrole dostopa so velikokrat pomanjkljivo ali celo narobe nastavljeni, ker je konfiguracija le teh pogosto predmet (ne nujno najboljše) subjektivne ocene nadzornikov omrežja.
V tej nalogi se bomo omejili na na tisto definicijo tega orodja, ki predpostavlja, da je to orodje, ki samo beleži in analizira mrežni promet in v primeru zaznave napada generira temu primeren alarm, ki je lahko tudi lažen, kar je še dodatna omejitev tovrstnih orodij.

3.5.2 Konkretne omejitve

Po pregledu splošnih omejitev orodij za detekcijo napadov si poglejmo še nekaj konkretnih omejitev in problemov, ki se pojavljajo pri detekciji mrežnega napada. Kompleksnost danega problema je veliko večja kot naprimer detekcija kloniranega mobilnega telefona ali kaj podobnega. Danes uporabljena orodja so še vedno v relativno zgodnjih razvojnih fazah, z okoli 60% do 80% uspešnosti pravilne zaznave mrežnega napada.

Poglejmo si nekaj konkretnih razlag omenjeih pomanjkljivosti:

- *Internet je okolje polno »šuma«, povzročenega tako na vsebinskem kot na paketnem nivoju.* Veliko število naključne aktivosti, ki nenadzorovano niha, lahko povzroči dobršno mero lažnih alarmov. Reportaža Bellowinove [17] analize poroča, da je velika mera *»škodljivih«* paketov povzročena zaradi napak v programski opremi mrežnih servisov. Ostali pa so povzročeni kot rezultat napake zastarelih ali napačnih DNS odgovorov ali kaj podobnega.

- *Malo število napadov.* Če predpostavimo, da se v praksi pojavi po deset napadov na vsak milijon internetnih povezav, kar je po vsej verjetnosti pretiravanje, in da ima sistem za detekcijo napadov zelo nizko stopnjo javljanja lažnih alarmov, recimo 0.1%, še vedno dobimo za razmerje lažnih in realnih alarmov 1:100. Na splošno je to situacija, kjer imamo pogostost opazovanega signala tako močno pod nivojem šuma, da obstaja zelo velika verjetnost, da ga dostikrat spregledamo.

- *Veliko mrežnih napadov je specifičnih za dano verzijo napadenega servisa, kjer je potrebno razlikovati od napadov na starejše verzije enakega servisa.* Zatorej je potrebno redno spremljati razvoj novih tehnik napadov ter bazo podpisov temu primerno redno vzdrževati in posodabljati.

- *Komercialne združbe in podjetja velikokrat kupijo orodja za detekcijo napadov samo zato, da izpolnjujejo varnostne kriterije in zahteve samo na papirju, in sicer samo tiste,* ki so zahtevani iz strani razninh zavarovalnic in/ali konzultantov.

- *Analiza šifriranega prometa, kot naprimer SSL kodirane spletne povezave, je praktično nemogoča.* V teoriji je seveda možno na robnih točkah varovanega omrežja vhodni mrežni promet dekriptirati, ga preveriti ter nato spustiti do

naslovljenih strežnikov, vendar bi za to potrebovali privatne ključe vseh uporabnikov varovanega omrežja, kar je pa samo po sebi nevarno in v praksi tudi zelo težko izvedljivo.

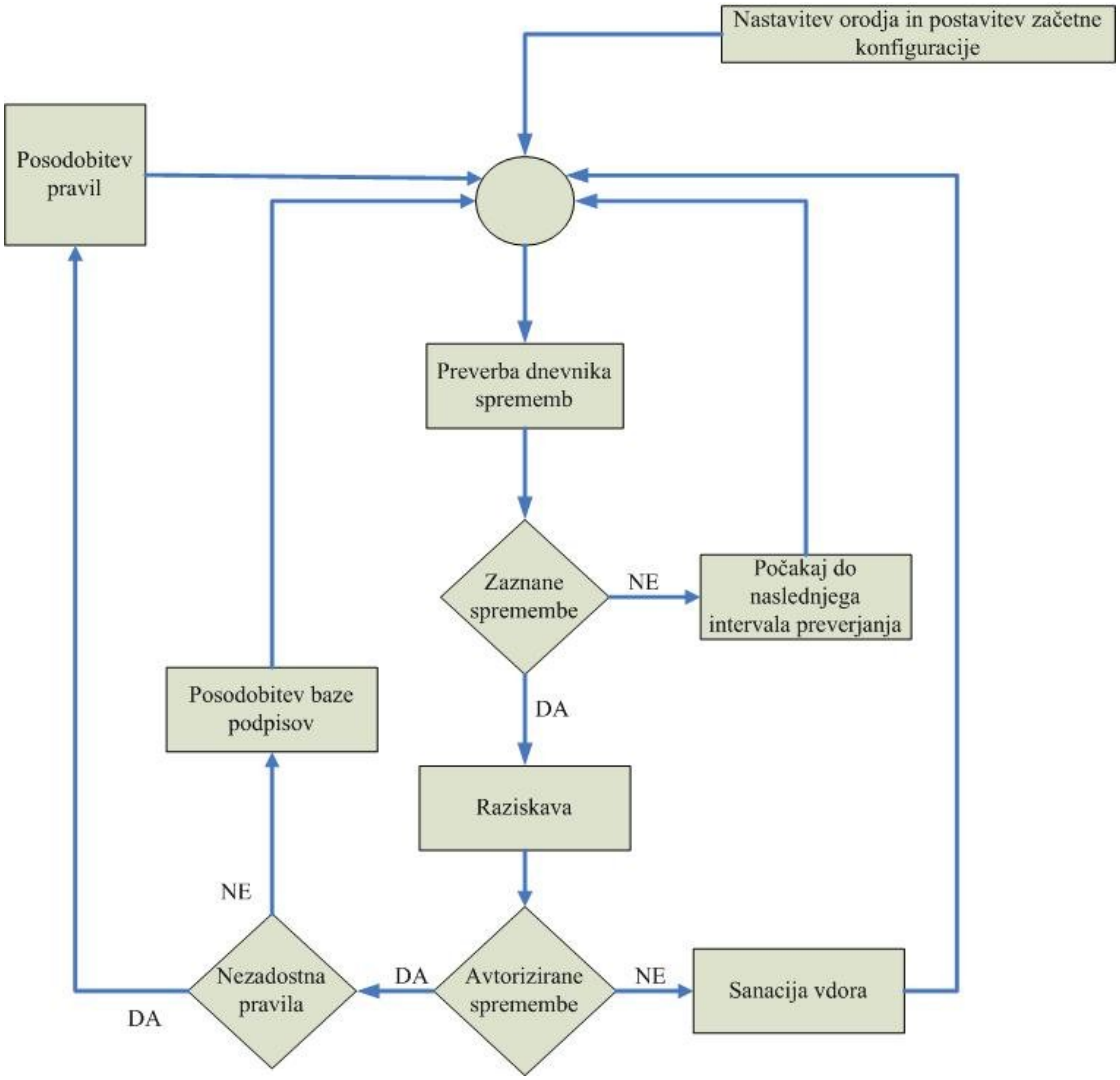
- *Večina omejitev iz konteksta požarnih zidov je problematična tudi v domeni orodij za detekcijo napadov.* Filtriranje na nivoju paketov je hitra in nepožrešna operacija, vendar relativno preprosto obvladljiva za namene zmanjšanja verjetnosti uspešne detekcije napada. Če napadalec uporabi fragmentacijo paketov, bi jih bilo za uspešno detekcijo potrebno sestaviti pred analizo, kar je pa v praksi zahtevnejša in požrešnejša operacija in kot taka neprimerna za večino omrežij, posebej še za tista, kjer je propustnost omrežij kritičnega pomena.

3.6 UPORABLJENA ORODJA

3.6.1 TripWire

TripWire je primer orodja za detekcijo vdorov, ki deluje na nivoju operacijskega sistema. Konkretneje na nivoju datotečnega sistema, z namenom ščititi integriteto datotek, tako da hrani podatkovno bazo podpisov varovanih datotek, katera se potem v rednih intervalih preverja, če je še vedno enaka prvotni različici. V primeru nepooblaščene spremembe ščitene datoteke se sproži alarm.

Pravila in načini ščitenja so nastavljivi v konfiguracijski datoteki, kjer se da izbirati in nastavljati različne tipe opazovanih nepravilnosti in resnost le teh, kar omogoča nadzorniku sistema nadzor nad datotečno aktivnostjo ter analizo vzroka neavtoriziranih dostopov ali sprememb sistemskih datotek. Nastavljena pravila nudijo kriterij za ločevanje normalne sistemske aktivnosti od nepravilnosti. Na ta način lahko nadzornik sistema, v primeru lažnih alarmov ali spremembe vzorcev delovanja sistema tudi posodobi konfiguracijsko datoteko in upošteva te spremembe.



Slika 7. Diagram različnih stanj orodja TripWire

Zgornji diagram različnih stanj orodja nam pokaže načine uporabe le tega. Kot je prikazano na vhodu v shemo, na začetku nadzornik namesti, nastavi in zažene orodje. Na uporabnikovo zahtevo se sproži preverjanje integritete varovanih datotek, kjer se orodje sprehodi čez datotečni sistem ter preveri ujemanje podpisa posamezne datoteke z ustrezno vrednostjo iz baze podpisov. V primeru neujemanja mora nadzornik preiskati vzroke spremembe datoteke in ugotoviti, ali gre za veljavno sistemsko aktivnost ali za neavtoriziran poseg. V primeru slednjega je potrebno sanirati nelegalno aktivnost oziroma če je bil alarm sprožen na osnovi legalne sistemske aktivnosti, mora nadzornik popraviti bodisi konfiguracijo orodja bodisi bazo podpisov.

TripWire je sistem za odkrivanje napada na nivoju operacijskega sistema. Prednost takega tipa orodja je v tem, da za delovanje ne analizira in ne išče vzorcev (podpisov) napada v opazovanem podatkovnem (ponavadi mrežnem) toku, temveč samo preverja spremembe na datotečnem sistemu glede na nastavljena pravila. Standardna konfiguracijska datoteka je na voljo že ob sami namestitvi orodja, katero lahko nadzornik sistema naknadno dopolni glede na specifične potrebe varovanega sistema.
Torej, TripWire orodje uporablja tehniko preverjanja integritete opazovanih komponent varovanega računalniškega sistema. Omenjena metoda zajema preverjanje

sprememb sistema, datotek, programov in strojne opreme, katere se v normalnih ali predvidenih okvirih delovanja ne pojavljajo. Orodje uporablja podpise, generirane po MD5 kriptografski metodi, katere hrani v varovani bazi in katera so izhodišče ali kriterij za vsa nadaljna preverjanja sistema. Naj omenimo še, da je za hranjenje baze podpisov zahtevano, da je na nekem drugem, ločenem sistemu z namenom onemogočiti (ali vsaj otežiti) napadalcu modificiranje le te.

3.6.2 Snort

Snort je oprtokodno orodje za detekcijo mrežnih vdorov z analizo mrežnega prometa za znanimi vzorci napadov. V določenih primerih je uporaben tudi za odkrivanje skeniranja omrežja ter poskusov preplavitve pomnilnika. Vse zaznane poizkuse napadov beleži ter omogoča opozarjanje administratorja preko elektronske pošte.

Poleg detekcije mrežnih napadov je lahko uporabljeno tudi v preventivne namene, saj nadzorniku omogoča konfiguracijo, na osnovi katere prepreči ali prekine povezavo, katera ne izpolnjuje kriterije normalnih okvirov delovanja. Za predstavitev zajete aktivnosti imamo na voljo širok spekter pripomočkov, kot so SnortSnarf, sguil, OSSIM in BASE …, kateri nam nudijo grafično predstavitev logiranih napadov.

Na voljo imamo dva načina delovanja: prva preprostejša možnost uporabe je samo zajem mrežnega prometa in shranjevanje mrežnih paketov na disk, medtem ko nam drug način delovanja nudi še izvajanje vsebinske analize zajetih paketov.

Za namene diplomske naloge je bilo orodje Snort uporabljeno na treh opazovanih računalniških sistemih, kar je podrobneje z rezultati opisano v nadaljevanju naloge.

3.6.3 Simx

Je (prototipna) verzija visoko interaktivnega raziskovalnega muholovca, razvitega za namene diplomske naloge (okoli 10.000 vrstic izvorne kode) [v]. Zasnovan je tako, da nudi servisne storitve operacijskega sistema brez vmesnih simulacijskih plasti. Ciljna platforma za uporabo tovrstnega orodja so že vnaprej rezervirani ter ločeni segmenti mreže, kateri ne nudijo nobenega realnega servisa ter nimajo nobene ciljne vrednosti, razen raziskovalne naloge. Po tej definiciji so torej vse zajete transakcije, vsi poskusi prijave v tak sistem ter vsi dostopi do podatkov na takem sistemu neavtorizirane narave.

Več o razvitem muholovcu je v nadaljevanju, kjer so predstavljene tehnične podrobnosti orodja, napisano tako, da lahko poleg predstavitve v nalogi služijo tudi kot specifikacija, na podlagi katere je možno nadaljevati razvojno aktivnost na zaenkrat še prototipni verziji orodja. V nadaljevanju praktičnega dela so tudi predstavljeni rezultati zajete aktivnosti z razvitim orodjem, katero je bilo postavljeno na enega izmed računalniških sistemov, ki sestavljajo testni poligon diplomske naloge.

4 TEORETIČNI PREGLED KONCEPTA MUHOLOVEC

4.1 ZGODOVINA MUHOLOVCA

Idejno ozadje koncepta muholovec je bilo prvič opisano v knjigi Clifford Stolla leta 1990 [8]. Knjiga je napisana kot novela, zasnovana na resnični zgodbi, ki se je zgodila avtorju. Na delovnem mestu je namreč odkril računalniški sistem, v katerega je nekdo že vdrl, nakar se je avtor odločil, da se bo poizkusil naučiti čim več o vdiralcu in samem načinu vdora. Z namenom izslediti vdiralčev izvor je Stoll na tem sistemu nastavil lažno okolje, da bi zaposlil vdiralca ter na ta način pridobil malo časa. Ideja lažnega okolja je bila slediti vdiralčeve povezave, medtem ko se le ta ukvarja z nastavljeno pastjo. Takrat ta rešitev še ni bila imenovana muholovec.

Leta 1999 je bila ta ideja uporabljena kot osnova za delo skupine Honeypot project [8], katere ustanovitelj in voditelj je Lance Spitzner. V naslednjem desetletju je skupina objavila veliko teoretičnega in praktičnega dela na temo muholovcev ter razvila in predstavila tehnike za uspešno postavitev in uporabo računalniških pasti z uporabo konceptov muholovec. Honeypot project je neprofitna raziskovalna organizacija računalniških strokovnjakov, specializiranih za računalniško varnost.

Knjiga “Honeypots, Tracking Hackers” [9], katere avtor je Lance Spitzer, podaja temeljne poglede na koncepte in arhitekturo muholovcev ter je kompetenten vir dotične terminologije in definicij, vezanih na tovrstno tematiko. Še vedno pa ostaja neznanka, kdo je prvi uporabil termin muholovec za opis računalniške pasti. V Spitzer-jevi knjigi najdemo opise prvih muholovec sistemov, od katerih pa ni bil nobeden od teh poimenovan s tem imenom.

4.2 UPORABNOSTNI VIDIKI MUHOLOVCA

Muholovci so izredno fleksibilno orodje s širokim naborom uporabnosti. Lahko si jih predstavljamo kot eno izmed orodij, ki jih imamo na voljo v arzenalu orodij za računalniško zaščito, katero po potrebi prilagodimo zahtevni situaciji. V splošnem jih lahko razdelimo na dve glavni kategoriji: komercialni izdelki z ozko definirano namenskostjo ter raziskovalni modeli. Komercialni izdelki so večinoma nizko interaktivni muholovci, uporabljeni kot sestavni deli vnaprej definiranega modela pasti, medtem ko so visoko interaktivni muholovci večinoma uporabljeni v raziskovalne namene. Kar pa še ne pomeni, da je temu vedno tako. Oba tipa muholovcev sta lahko uporabljena tako za *produkcijske* kot *raziskovalne* namene. Prav tako ne obstaja kriterij, po katerem bi lahko rekli, da je ena kategorija boljša od druge, temveč je namen tovrstne kategorizacije zgolj pomoč pri identifikaciji cilja, katerega hočemo z uporabo muholovca doseči. Se pravi, da so pri prvi kategoriji muholovci uporabljeni za enega izmed treh možnih ciljev organizacije: *preprečevanje napadov*, *detekcija napadov* ter *odgovor na napade*. Vsak izmed zgoraj naštetih ciljev predstavlja različno vrednost za različne organizacije. Medtem ko so nekatere ustanove zainteresirane za samo študijo in trende različnih napadov, imajo lahko druge namen postaviti sistem za zgodnejše opozarjanje napadov itd. V nadaljevanju si bomo pogledali podrobnejšo predstavitev uporabnosti različnih tipov rešitev z uporabo muholovcev.

4.2.1 Preprečevanje napadov

Muholovci lahko pomagajo preprečiti napade na več načinov. Recimo, lahko nam pomagajo pri preprečevanju avtomatiziranih napadov iz strani raznih črvov in računalnikov zombijev, povezanih v mrežo. Tovrstni napadi so zasnovani na naključnem skeniranju celotnih mrežnih segmentov in iskanju ranljivih sistemov. Ko se tak sistem najde, se ga izkoristi za vdor ter nadaljno replikacijo črva na tak sistem, kateri je potem uporabljen za nadaljna skeniranja. Eden izmed možnih načinov obrambe pred takimi napadi, ki jih nudijo muholovci, je upočasnitev procesa iskanja ranljivih tarč. Tak tip muholovca imenujemo tudi "lepljivi muholovec", katerega se ponavadi nastavi za opazovanje neuporabljenih delov IP naslovnega prostora. Ko se detektira poizkus skeniranja na takem segmentu omrežja, se tovrstni muholovec poskuša z uporabo različnih tehnik na transportnem (TCP) nivoju mrežnega sklada napadalca upočasniti. Primer take tehnike je uporaba ničelne vrednosti za širino drsečega okna, kar postavi napadalca, seveda ob predpostavki, da se le ta drži standardov kontrole prenosa TCP protokola, v čakajoče stanje. Opisana taktika nam lahko pomaga preprečiti ali znatno upočasniti razširjanje mrežnih črvov, kateri so že vdrli v notranje mrežne segmente. Kot primer tovrstnega orodja si lahko pogledamo La Brea Tar pit [2]. "Lepljivi muholovci" spadajo v kategorijo nizko interaktivnih muholovcev. Nekateri jih imenujejo kar ne-interaktivne rešitve, saj nudjio samo upočasnitev širitve avtomatiziranih napadov.

Muholovci so uporabni tudi za zaščito pred neavtomatiziranimi napadi. Koncept je zasnovan na prevari ali zavajanju. Namen je zmesti napadalca in ga pripraviti do tega, da trati čas z ukvarjanjem s tovrstno pastjo, medtem ko pridobimo na času za uspešno detekcijo napadalčeve aktivnosti in za pripravo temu primernega odgovora za ustavitev le te. Lahko si zamislimo tudi sledeč scenarij: recimo, da je splošno znano dejstvo, da naša organizacija uporablja muholovce. Ker napadalec ne ve, kateri sistemi so pravi in kateri nastavljeni, se lahko, če je pazljive narave, tudi premisli in naše organizacije raje ne napade. Kot primer tovrstnega orodja si lahko pogledamo Deception Toolkit [5].

4.2.2 Zaznavanje napadov

Drugi namen uporabe muholovec orodij je detekcija napadov na varovano računalniško omrežje. Odkrivanje napada je kritičnega pomena, saj mora znati ločiti med dejanskim napadom ali sistemsko okvaro na enemi izmed sestavnih delov mrežnega segmenta. Tovrstne napake so in bodo vedno prisotne, ne glede na stopnjo zaščite pri danem računalniškem omrežju. Detekcijo napada nam omogoči hiter odgovor na napad ter zmanjševanje posledic le tega.

Detekcija se je sčasoma izkazala za zelo zahtevno nalogo in različne tehnologije v obliki detektorjev napada, sistemskih senzorjev in različnih logerjev so se izkazali kot neefektivne iz večih razlogov: generirajo zelo veliko količino podatkov, od katerih je za nameček še velik odstotek lažnih alarmov, nezmožnost detekcije novih tipov napada, velika večina teh orodij ni primerna za delo v okoljih, kjer so podatki šifrirani ter še ni prirejena za IPv6 protokol. Muholovci uspešno odpravljajo večino teh pomanjkljivosti, saj zajemajo omejene količine podatkov, kateri so po definiciji neavtorizirana aktivnost, uspešno beležijo nove tipe napadov ter načeloma nimajo omejitev za delo v okoljih s šifriranimi podatki ali IPv6 okoljih.

4.2.3 Odgovor na napade

Muholovci nam nudijo tudi obrambo v obliki odgovora na napade. Tukaj se pojavlja vprašanje, kaj naj sploh naredi organizacija po odkritju napada? Nemalokrat so informacije o identiteti, motivih napadalca, povzročeni škodi in tehnikah vdora zelo omejene ali jih pa sploh ni, vendar je v takih situacijah potreba po tovrstnih informacijah kritičnega pomena. Izpostavimo samo dva večja problema pri iskanju primernega odgovora na že udejanjen napad:

- Kot prvo izpostavimo problem analize napadenega sistema, katerega ponavadi ne moremo preprosto izklopiti. Kot primer si vzemimo teoretično situacijo napada na poštni strežnik neke organizacije. Po detekciji napada na tak strežnik si velikokrat ne moremo privoščiti izklop le tega za nadaljno forenzično analizo, ker imamo v ozadju veliko množico uporabnikov, katerim hočemo zagotoviti čim bolj nemoteno delovanje.
- Kot drugi večji problem analize po napadu pa izpostavimo veliko količino podatkov, katere je potrebno analizirati in ločiti napadalčevo aktivnost od legalne aktivnosti na danem sistemu, kar nemalokrat spominja na iskanje igle v kopici sena.

Muholovci ponujajo rešitve za oba izpostavljena problema, na katera naletimo po vdoru. Računalniške sisteme z muholovcem lahko preprosto in brez vpliva na primarno servisno dejavnost organizacije izklopimo za nadaljno forenzično analizo. Poleg tega nam analiza zajete aktivnosti na takem sistemu ne predstavlja večjih problemov pri ločevanju napadalčeve aktivnosti od legalne, saj je po definiciji vsa aktivnost na danem sistemu neavtorizirana in kot taka predmet nadaljne raziskave. Če sedaj povzamemo poglavitne prednosti muholovcev, lahko vidimo, da nam omogočajo zelo poglobljen vpogled v napadalčevo aktivnost (kdo, kaj, kako in s kakšnimi orodji), kar v splošnem pripomore k učinkovitejšemu odgovoru na napad in zaščiti sistema v prihodnje.

4.2.4 Uporaba muholovcev v raziskovalne namene

Kot smo že prej omenili, so lahko muholovci uporabljeni v raziskovalne namene, kjer nas zanima poglobljen vpogled v potencialne grožnje, kar pa je primarna naloga muholovcev, kajti glavni problem, s katerim se soočajo strokovnjaki na področju računalniške varnosti, je pomanjkanje informacij o grožnjah. Povedano drugače: muholovci nam nudijo podatke, ki nas branijo pred neznanim napadom. Zbrani podatki so namenjeni nadaljni analizi v različne namene, kot so analiza trendov, identifikacija novih metod in orodij, uporabljenih za napad, identifikacija napadalcev in njihovih motivov ter postavitev sistemov za zgodnje opozarjanje.

4.3 PREDNOSTI IN SLABOSTI

V nadaljevanju si poglejmo nekaj prednosti in slabosti značilnih muholovcev:

- *Muholovci zajemajo zelo omejene (ponavadi male) količine podatkov.* Ker beležijo podatke le v primeru napadalčeve aktivnosti na nadzorovanem sistemu, so te količine manjšega obsega, vendar zelo dragocene iz stališča nadaljne analize le teh. Načeloma manjša in kontrolirana vsebina zajetega nam zelo olajša upravljanje ter nadaljno analizo zajete aktivnosti.
- *Muholovci zmanjšajo število lažnih alarmov.* Eden večjih izzivov večine orodij za detekcijo napadov je količina lažnih alarmov, ki jih tovrstna orodja

sproducirajo. Tukaj lahko kot primer potegnemo analogijo s sorodnim problemom avtomobilskih alarmov, katere se vgrajuje v avtomobile, z namenom preprečitve kraje le teh. Ker pa se avtomobilski alarmi zelo pogosto sprožajo iz napačnih razlogov, so se ljudje praktično navadili, da jih preprosto ignorirajo. Pomislite, kaj bi storili, če bi slišali avtomobilski alarm, medtem ko bi hodili po parkirišču? Najverjetneje nič. Podobna problematika se pojavlja pri večini današnjih orodij za detekcijo napadov. Namreč večja kot je verjetnost lažnih alarmov pri dani varnostni tehnologiji, manjša je uporabnostna vrednost takega orodja. Muholovci dramatično zmanjšajo število lažnih alarmom, ker je vsa zajeta aktivnost na takem sistemu po definiciji neavtorizirana.

- *Muholovci zajamejo nove, še nepoznane vrste napadov.* Poleg lažnih alarmov je to drugi največji problem običajnih orodij za detekcijo napadov, kateri so v veliki večini primerov nesposobni zaznati novo, še neznano vrsto napada. To pa je tudi največja razlika med muholovci in tradicionalnimi varnostnimi orodji, katera slonijo večinoma na detekciji že znanih (statističnih) vzorcev napada, ker mora vedno nekdo najprej biti napaden, da se nova tehnika napada detektira in objavi preden se vgradi novo detektirani vzorec med statistične podatke za nadaljno obrambo. Muholovci nam pa po drugi strani nudijo ravno to, saj je vsa zajeta aktivnost na računalniškem sistemu z muholovcem neavtorizirana, kar posledično izpostavlja tudi nove ali še ne znane tehnike napada.
- *Muholovci znajo zajeti šifrirano aktivnost.* Ker vedno več organizacij uporablja kriptirane kanale (kot so Secure Shell [SSH], IP Securiy Protokol [IPsec] ter Secure Sockets Layer [SSL]) za komunikacijo v svojih okoljih, bi to lahko predstavljalo problem za konvencialna orodja, ki analizirajo mrežni promet. Ker pa muholovci delujejo na končni točki take komunikacije, velikokrat v samem konekstu jedra operacijskega sistema, kjer je promet že dekriptiran, nam to ne predstavlja problema.
- *Muholovci delujejo z IPv6.* Večina muholovcev je zasnovana tako, da delujejo v vseh IP okoljih, vključno z IPv6, kateri je najnovejša verzija IP protokola in katerega mnoge organizacije, kot so naprimer Ameriško ministrstvo za obrambo ter mnoge vladne ustanove, že na široko uporabljajo. Mnoga sorodna orodja, kot so naprimer požarni zidovi ali detektorji vdorov, še niso dodobra pripravljeni na IPv6.
- *Muholovci nam nudijo visoko mero fleksibilnosti.* So zelo prilagodljvo orodje, katero se da uporabiti v širokem spektru različnih okolij: od nastavljenih polj v podatkovnih bazah pa vse do celotnih segmentov računalniške mreže, katera je postavljena z namenom, da se vanjo vdre.
- *Muholovci zahtevajo minimalne zahteve za svoje delovanje.* To drži tudi za večja omrežja. Že zelo zastareli računalnik je lahko postavljen v vlogo kontrolne točke, iz katere je omogočen nadzor celotnih segmentov mreže.
- *Muholovci imajo omejeno polje delovanja ali zajemanja napadalčeve aktivnosti.* Lahko beležijo napadalčevo aktivnost na sistemu samem, ne morejo pa zajeti aktivnosti na ostalih oddaljenih sistemih, dokler se sama aktivnost ne *dotakne* sistema, na katerem je muholovec nastavljen. Za premostitev te ovire imamo na voljo množico načinov, kako privabiti napadalčevo aktivnost, kot so

na primer na videz zanimive datoteke, ki nimajo dejanske vrednosti ali podobne rešitve.

- *Tveganje.* Vsakič, ko uvajamo kakšno novo tehnologijo, imamo opravka z določeno mero tveganja. Na primer, pri postavljanju računalniških sistemov z muholovcem vedno obstaja tveganje, da napadalec prepozna nastavljeno vabo, nas prelisiči ter prevzame nastavljen sistem. Nakar lahko tak sistem uporabi za nadaljni napad na bodisi notranje ali zunanje tarče. Za primerjavo omenimo, da so lahko celo konvencionalna orodja, kot je na primer Snort [6], ranljiva na oddaljene napade. Muholovci niso izjema.

4.4 RAZLIČNI TIPI MUHOLOVCEV

4.4.1 Visoko interaktivni muholovci

Za boljše razumevanje muholovcev, jih lahko razdelimo na dve glavni kategoriji: muholovci z visoko in nizko stopnjo interaktivnosti. Kaj to pomeni? *Interaktivnost je stopnja aktivnosti, ki je dovoljena napadalcu na računalniškem sistemu z nastavljenim muholovcem.* Večja kot je stopnja interaktivnosti, toliko več lahko napadalec uporablja (in izrablja) nastavljen sistem. Temu primerno se tudi poveča stopnja tveganja, vendar posledično tudi večja količina zajete aktivnosti, pripravljene za nadaljno analizo. Ne glede na to, da nam opisane kategorije omogočajo zelo grobo tipiziranje muholovcev, nam olajšajo razumevanje tega, relativno novega koncepta, njegovih zmožnosti ter omejitev.

4.4.2 Nizko interaktivni muholovci

Nizko interaktivni muholovci večinoma samo oponašajo različne operacijske sisteme in storitve. Napadalčeva aktivnost je omejena na okvirje in zmožnosti, katere oponašani servis ponuja. Kot primer tovrstnega orodja si lahko pogledamo *BackOfficer Friendly [3]*, ki je zelo preprosta muholovec rešitev, katera nam nudi oponašanje sedmih različnih mrežnih servisov. Napadalci so na takšnih sistemih zelo omejeni, saj imajo po prijavi v tak sistem na voljo izvajanje omejenega nabora (preprostih) ukazov.

So pa zato nizko interaktivna muholovec orodja preprostejša za namestitev in uporabo kot muholovci z višjo stopnjo interaktivnosti. Ponavadi imamo na voljo množico nastavitev, katere so ponujene na voljo končnemu uporabniku, se pravi skrbniku mreže z muholovec računalniškimi sistemi. Do željenega oponašanja servisa ali operacijskega sistema nas ponavadi loči samo par klikov. Kot primer tovrstnega orodja si lahko pogledamo *Specter [4]*, ki je komercialno orodje, namenjeno operacijskim sistemom Windows. Lahko simulira do 13 različnih operacijskih sistemov ter nadzira 14 različnih servisov. Tovrstni uporabniški vmesniki so zasnovani za preprosto uporabo, kjer lahko večino zahtevanega dosežemo z nekaj kliki.

Kot smo že omenili, nizko interaktivni muholovci predstavljajo relativno malo tveganja, saj nudijo napadalcu zelo omejeno aktivnost in dostop do sistema. Dostopa do samega operacijskega sistema praktično ni, napadalec nima možnosti niti mesta nalagati svoja orodja. Prav tako ne obstajajo storitve, preko katerih se bi dalo vdreti v sistem.

Kot vidimo zgoraj, nam tesno zaprti muholovci zmanjšujejo stopnjo izpostavljenosti ter posledično tveganje, vendar je to samo ena plat kovanca. Po drugi strani je jasno,

da bolj ko omejimo napadalčevo aktivnost, bolj omejimo ali zmanjšamo obseg zajetih podatkov.

4.4.3 Prednosti in pomanjkljivosti nizko interaktivnih muholovcev proti visoko interaktivnim

Nizko interaktivni muholovci (Simulacija operacijskega sistema in njegovih servisov)

- Preprosta namestitev in uporaba; največkrat je zahtevana samo še osnovna konfiguracija po začetni inštalaciji
- Manjše tveganje, ker nam simulirani servisi sami po sebi definirajo in s tem omogočijo kontrolo nad dostopom, katerega želimo omogočiti napadalcu
- Zajem omejene količine podatkov, v glavnem same transakcije in zelo omejena interaktivnost.

Visoko interaktivni muholovci (Brez simulacije; uporaba realnega operacijskega sistema in njegovih servisov)

- So zahtevnejši za namestitev in samo uporabo (komercialne različice se nagibajo k preprostejši uporabi)
- Lahko zajamejo mnogo več podatkov kot nizkoodzivni muholovci, kot so naprimer: nove, še neznane verzije napadalčevih orodij, njegove mrežne komunikacije pa vse do zajema napadalčeve aktivnosti na terminalih.
- Povečana stopnja tveganja zaradi omogočanja uporabe realnega operacijskega sistema napadalcu. Različne organizacije imajo različne motive in cilje ter posledično uporabljajo različne vrste muholovcev. Skupni imenovalec na splošno pa je, da razne komercialne organizacije (banke, proizvodni obrati ter prodajalne mreže) raje uporabijo nizko interaktivne verzije muholovcev, zaradi preprostejše namestitve in uporabe le teh. Medtem ko so visoko interaktivni muholovci večinoma uporabljeni pri organizacijah z zelo specifičnimi potrebami, mnogokrat za raziskovalne namene. Primeri takšnih organizacij so lahko vojska, vladne ustanove ter izobraževalne ustanove.

5 PROTOTIP RAZISKOVALNEGA MUHOLOVCA: SIMX

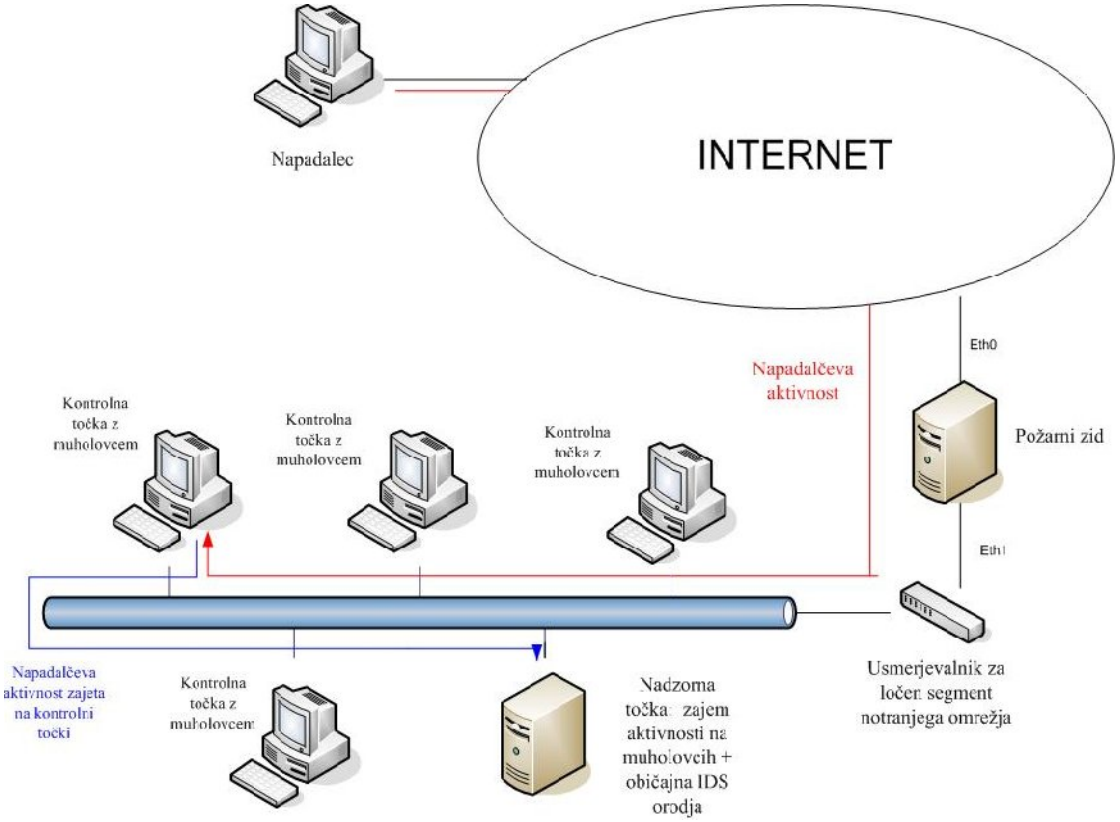
Je prototipna različica visoko interaktivnega raziskovalnega muholovca, zasnovanega tako, da nudi servisne storitve operacijskega sistema brez vmesnih simulacijskih plasti.

Ciljna platforma za uporabo tovrstnega orodja so že v naprej rezervirani ter ločeni segmenti mreže, kateri ne nudijo nobene realne (ali produkcijske) storitve ter nimajo nobene ciljne vrednosti razen raziskovalne naloge. Po tej definiciji so torej vse zajete transakcije, vsi poskusi prijave v tak sistem ter vsi dostopi do podatkov na takem sistemu neavtorizirane narave. Primer uporabe *simx* muholovca je lahko namestitev na spletni ali datotečni strežnik, kateri se nahaja v ločenem segmentu mreže in je namenjen izključno raziskovalnim namenom. Tako da je aktivnost, zajeta na takšnem sistemu, neavtorizirana in škodljiva po definiciji.

Zajete informacije o aktivnostih na takšnih sistemih se potem pošljejo preko prikritega kanala nazaj skrbniku v analizo za različne namene, kot so: analiza trenutnih trendov, identifikacija novih orodij in metod za vdore, identifikacija samih napadalcev in njihovih skupnosti, zagotavljanje zgodnjega opozarjanja, predvidevanja in samega razumevanja napadalčevih motivov.

5.1 OPIS FUNKCIONALNOST

Simx je orodje za zajem podatkov, zasnovano tako, da brez napadalčeve vednosti zajame čim obširnejšo aktivnost na kontrolni točki in le to potem na administratorjevo zahtevo pošlje na centralno točko preko poljubnega prikritega kanala.



Slika 8. Arhitektura muholovca *simx*

Kategorije zajete aktivnosti:

- aktivnosti na vseh konzolah tipa (prestrezanje funkcionalnosti tty gonilnika, kar nam omogoči zajem napadalčeve aktivnosti na tipkovnici):
    - tty
    - pts
    - pty
    - ptm
    - cua
    - console
- dostopi do datotek (prestrezanje sistemskih klicev, kar nam omogoči zajem napadalčeve aktivnosti na datotečnem sistemu):
    - open dogodek
    - read dogodek
    - write dogodek
- detekcija preverjanja odprtih vrat (samo transportni nivo):
    - SYN
    - NULL
    - FINN
    - ACK
    - Window
    - Xmas
    - Maimon

Na zahtevo administratorja omrežja se vse informacije o zajeti aktivnosti preko skritega kanala pošljejo na nadzorno točko za nadljno analizo napadalčeve aktivnosti. Na voljo imamo različne (transportne) protokole:

- icmp
- udp
- tcp

Z namenom, da ostanemo čim bolj neopaženi tako na samem računalniškem sistemu kot na mreži, je transportna aktivnost skrita tako, da vse mrežne pakete (ukazne in podatkovne) označimo z ustreznimi ID polji, katera so kriteriji za skrivanje le teh pred ostalimi (napadalčevimi) aplikacijami.

5.1.1 Arhitekturni presek orodja

Na grobo ga lahko razdelimo na dve glavni komponenti:

- Prva je gonilnik, katerega namestimo na poljubne kontrolne točke. Služi nam za nadzor napadalčeve aktivnosti (na terminalih, uporabljene datoteke ter izhodni mrežni promet), kar lahko potem skrbnik na zahtevo zbere na centralni točki za kasnejšo analizo.

- Druga komponenta je strežnik, kateri zbira podatke iz različnih kontrolnih točk. Ponavadi ga inštaliramo na ločen in zaščiten računalniški sistem.

5.1.2 Strežnik

Simx strežnik je centralna točka za zajem podatkov, zbranih na kontrolnih točkah, na katerih so inštalirani *simx* gonilniki za zajem napadalčeve aktivnosti. Namesti se ga na ločeno in zaščiteno točko v rezerviranem (v raziskovalne namene) segmentu mreže.

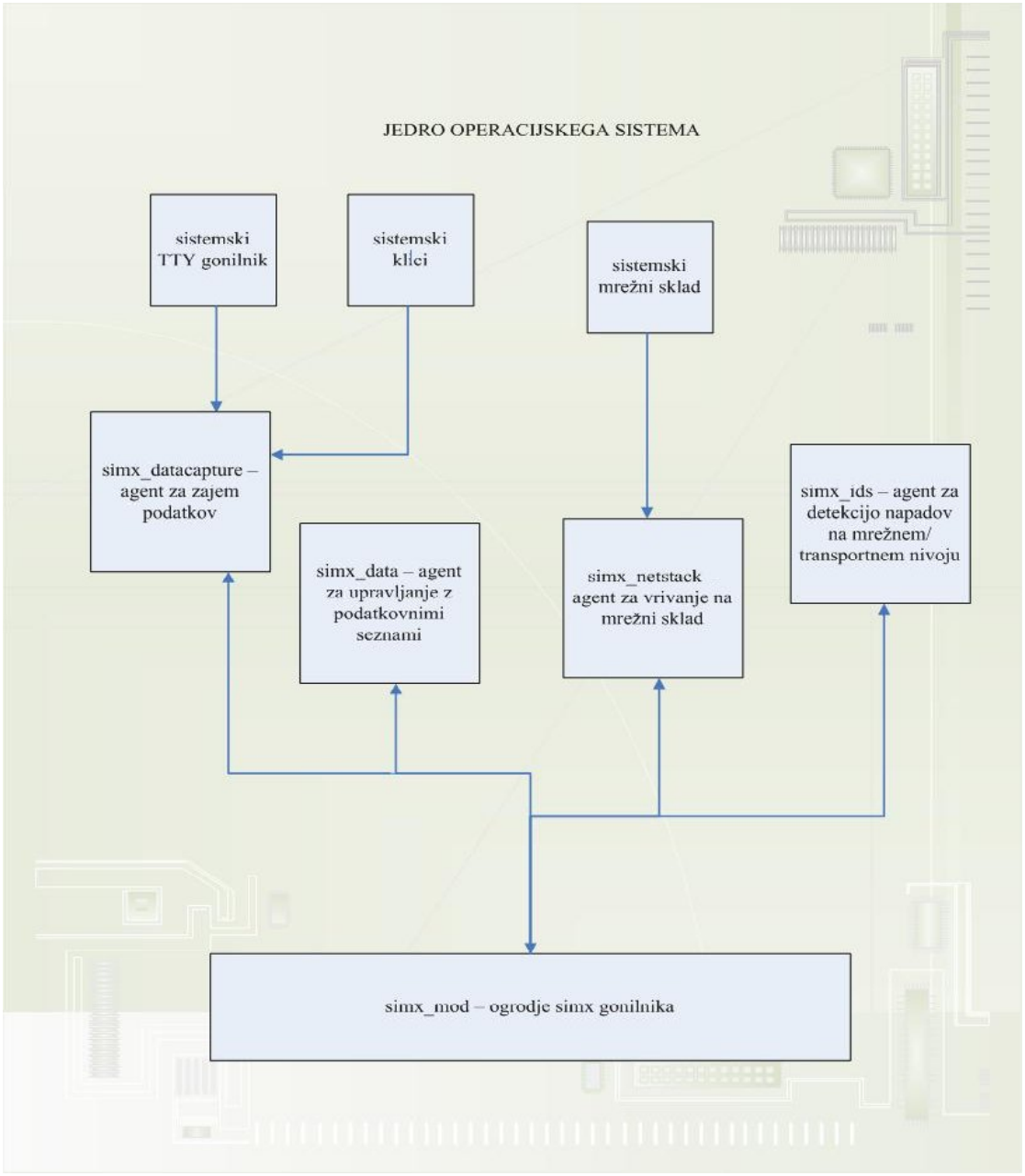
Napisan je v programskem jeziku C++ kot aplikacija, namenjena upravljanju, nadzoru in zbiranju podatkov iz kontrolnih točk opazovanega omrežja.

5.1.3 Gonilnik

Gonilnik za zajem napadalčeve aktivnosti je nameščen na vse kontrolne točke rezerviranega segmenta mreže.

Napisan je v programskem jeziku C kot gonilnik za operacijski sistem GNU/Linux in kot tak deluje v samem jedru operacijskega sistema. To nam omogoča zelo tesno integracijo z gostujočim sistemom, kot je na primer prestrezanje sistemskih klicev, nadzor aktivnosti na mrežnem skladu za analizo in manipulacija mrežnih paketov, skrivanje prisotnosti in aktivnosti …

Zajeta aktivnost se interno hrani v pomnilniku do zahteve po določenem tipu podatkov, nakar se zajeti podatki zapakirajo v mrežne pakete in pošljejo preko skritega kanala administratorju na kontrolno točko. Tehnike zajema napadalčeve aktivnosti, detekcije iskanja odprtih vrat in zagotavljanja neopaženosti na danem računalniškem sistemu so podrobneje opisane v nadaljevanju.



Slika 9. Arhitektura simx gonilnika

5.1.3.1 Tehnike zajema podatkov o napadalčevi aktivnosti

Razviti gonilnik je zaenkrat zasnovan tako, da nam omogoča zajem dveh tipov napadalčeve aktivnosti:

- Aktivnost na datotečnem sistemu

Zajem napadalčeve aktivnosti na opazovanem datotečnem sistemu je realiziran s prestrezanjem ustreznih sistemskih klicev, uporabljenih na VFS nivoju datotečnega sistema za vhodno/izhodne operacije nad datotekami.

Primer prestrezanja sistemskih klicev:

```
int32_t sx_syscall_hook_register() {
…
    unsigned long** syscall_table = NULL;
    syscall_table = get_syscall_table();
    if (!syscall_table) {
        return FAILURE;
```

```
    }
…
    unsigned long flags;
    spin_lock_irqsave(&sx_syscall_lock, flags);

    // save original system calls
    orig_sys_open = (int32_t (*)(const char *, int,
int))syscall_table[__NR_open];
    orig_sys_read = (ssize_t (*)(unsigned int,char
*,size_t))syscall_table[__NR_read];
    orig_sys_readv = (ssize_t (*)(unsigned int,const struct iovec *
,size_t))syscall_table[__NR_readv];

    // put our syscall wrappers in place
    syscall_table[__NR_open] = (unsigned long *)sx_sys_open;
    syscall_table[__NR_read] = (unsigned long *)sx_sys_read;
    syscall_table[__NR_readv] = (unsigned long *)sx_sys_readv;

    spin_unlock_irqrestore(&sx_syscall_lock, flags);
```

Izpis 5-1: Prestrezanje sistemskih klicev

- Aktivnost na terminalih

Zajem napadalčeve aktivnosti na opazovanih terminalih je realiziran s prestrezanjem ustreznih funkcijskih klicev različnih tipov tty gonilnika, katerega naloga je upravljanje tako lokalnih kot oddaljenih terminalskih aktivnosti. Razvita različica gonilnika uporablja tehniko, opisano v [23], kjer je opisano, kako se vriniti med tipkovnico in tty gonilnik in izvajati poljubne operacije glede na generirano aktivnost (v našem primeru gre za zajem napadalčeve aktivnosti na danem terminalu).

Primer prestrezanja aktivnosti tty gonilnika:

```
int32_t sx_tty_driver_open(struct tty_struct *tty, struct file *filp) {
…
    switch (tty->driver.type) {
        case TTY_DRIVER_TYPE_CONSOLE: {
            ret = orig_vc_open(tty, filp);
        } break;

        case TTY_DRIVER_TYPE_SERIAL: {
            ret = orig_serial_open(tty, filp);
        } break;

        case TTY_DRIVER_TYPE_PTY: {
            ret = orig_pty_open(tty, filp);
        } break;

        default: {
            DPRINT("ERROR: Unknown TTY driver type!");
            return FAILURE;
        } break;
    }

    if (ret >= 0) {

        unsigned long flags;
        spin_lock_irqsave(&sx_keylog_lock, flags);
        if (tty != NULL
            &&
              ((tty->driver.type == TTY_DRIVER_TYPE_CONSOLE &&
                TTY_NUMBER(tty) < MAX_TTY_CON - 1) ||
              (tty->driver.type == TTY_DRIVER_TYPE_PTY &&
               tty->driver.subtype == PTY_TYPE_SLAVE &&
```

```
                TTY_NUMBER(tty) < MAX_PTS_CON)
              )
             &&
                tty->ldisc.receive_buf != NULL
             &&
                tty->ldisc.receive_buf != sx_receive_buf) {

               sx_init_tty(tty, TTY_INDEX(tty));
          }
          spin_unlock_irqrestore(&sx_keylog_lock, flags);
    }
```

Izpis 5-2: Prestrezanje terminalske aktivnosti

5.1.3.2 Odkrivanje iskanja odprtih vrat

V jedru lahko z vrinjenjem na mrežni sklad danega operacijskega sistema prestrezamo ves mrežni promet. Kar v gonilniku simx izkoriščamo za dvoje namenov:

- Analiza mrežnega prometa, z namenom iskati značilne vzorce, ki se pojavljajo pri iskanju odprtih vrat. Podrobneje opisano v nadaljevanju.
- Prikrivanje našega mrežnega prometa, z namenom ostati čim bolj neopažen iz strani napadalca. Podrobneje opisano v naslednjem poglavju.

Razvita različica gonilnika simx je trenutno podprta samo na Linux jedru verzije 2.4.x, kjer imamo za vrivanje na mrežni sklad na voljo koncept netfilter hooks [22], kateri nam omogoča prestrezanje mrežnega prometa na različnih nivojih mrežnega sklada na poti ravnokar dobljenega paketa iz mreže pa vse do uporabniškega konteksta, v katerem tečejo navadne mrežne aplikacije.

Primer registriranja v mrežni sklad:

```
uint32_t hook_register() {
    uint32_t status = SUCCESS;
    // Incoming hook
    in_hook_g.hook     = sx_in_watch; // simx handler of incoming
network traffic
    in_hook_g.pf       = PF_INET;
    in_hook_g.priority = NF_IP_PRI_FIRST;
    in_hook_g.hooknum  = NF_IP_PRE_ROUTING;  // First incoming hook
    // Outgoing hook
    out_hook_g.hook     = sx_out_watch; // simx handler of outgoing
network traffic
    out_hook_g.pf       = PF_INET;
    out_hook_g.priority = NF_IP_PRI_FIRST;
    out_hook_g.hooknum  = NF_IP_POST_ROUTING; // Last outgoing hook
    // Register both
    sx_log_dbg(SX_DBG_NETFILTER, " registering incoming hook...");
    nf_register_hook(&in_hook_g);
    sx_log_dbg(SX_DBG_NETFILTER, " registering outgoing hook...");
    nf_register_hook(&out_hook_g);
    return status;
}
```

Izpis 5-3: Registracija na mrežni sklad

Zgoraj opisana metoda nam omogoči prestrezanje tako vhodnega kot izhodnega mrežnega prometa. Z analizo TCP zastavic vhodnega prometa lahko tako zaznamo več različnih tehnik uporabljenih pri iskanju odprtih vrat računalniškega sistema.

Poglejmo si nekaj primerov odkrivanja iskanja odprtih vrat na transportnem nivoju mrežnega sklada:

```
uint32_t sx_ids_check_detect_scan(struct sk_buff *a_skb) {
…
        case IPPROTO_TCP: {
            struct tcphdr* tcp_header = \
                (struct tcphdr*)((char*)a_skb->nh.iph + sizeof(struct
iphdr));
…
            // NULL scan
            if (!tcp_header->fin &&
                !tcp_header->syn &&
                !tcp_header->rst &&
                !tcp_header->psh &&
                !tcp_header->ack &&
                !tcp_header->urg &&
                !tcp_header->ece &&
                !tcp_header->cwr) {

                GET_TIME()
                sx_log_dbg(SX_DBG_NETFILTER,
                    " NULL scan detected from %s:%d to port %d",
                    src_ip_addr_str,
                    ntohs(tcp_header->source),
                    ntohs(tcp_header->dest));
…
             // FIN scan
             else if (tcp_header->fin &&
                !tcp_header->syn &&
                !tcp_header->rst &&
                !tcp_header->psh &&
                !tcp_header->ack &&
                !tcp_header->urg &&
                !tcp_header->ece &&
                !tcp_header->cwr) {

                GET_TIME()
                sx_log_dbg(SX_DBG_NETFILTER,
                    " FIN scan detected from %s:%d to port %d",
                    src_ip_addr_str,
                    ntohs(tcp_header->source),
                    ntohs(tcp_header->dest));

                rec.type = pst_FIN;
…
            // Xmas scan
            else if (tcp_header->fin &&
                tcp_header->psh &&
                tcp_header->urg) {

                GET_TIME()
                sx_log_dbg(SX_DBG_NETFILTER,
                    " Xmas scan detected from %s:%d to port %d",
                    src_ip_addr_str,
                    ntohs(tcp_header->source),
                    ntohs(tcp_header->dest));
```

Izpis 5-4: Odkrivanje iskanja odprtih vrat na transportnem nivoju

5.1.3.3 Zagotavljanje neopaženosti na opazovanem računalniškem sistemu

Ker je namen muholovca zajem napadalčeve aktvnosti, je potrebno čim bolj skriti prisotnost samega muholovca in njegove aktivnosti. V ta namen sem uporabil več tehnik:

- skrivanje gonilnika

Napadalec po vdoru v računalniški sistem ponavadi podrobno preveri prisotnost orodij, postavljenih za proženje alarmov in pobriše sistemske dnevnike, kateri bi morebiti razkrili njegovo prisotnost. Ker bi bil simx gonilnika na listi naloženih gonilnikov zelo sumljiv, sem se odločil uporabiti tehniko [24] vrivanja prevedene gonilniške kode v neki drugi, poljuben in ne vpadljiv tip gonilnika, kot je naprimer gonilnik za zvočno kartico *sound* (gostujoči gonilnik je poljuben).

- skrivanje mrežnega prometa

Ker gonilnik komunicira s kontrolno točko in na zahtevo generira mrežni promet za prenos zajetega materiala v nadaljno analizo, je potrebno tak promet skriti pred napadalcem.V ta namen sem za pošiljanje mrežnega paketa uporabil funkcionalnost mrežne naprave na najnižjem (še dosegljivem) nivoju mrežnega sklada, z namenom, da je poslani paket vidljiv na čim manjšem številu nivojev. Uporabljena tehnika pošlje paket neposredno gonilniku mrežne naprave ter tako zaobide ostale mrežne nivoje, kateri so med drugim vidni tudi zunaj jedrnega konteksta, to je mrežnim aplikacijam za zajem mrežnega paketa.

Poglejmo si primer pošiljanja mrežnega paketa:

```
// Send SX packet down the netstack
int _send_reply(struct sk_buff* a_sb) {

    sx_log_dbg(SX_DBG_NETFILTER, "Packet dump before sendof:");
    dump_packet(a_sb);

    // send reply

    if (dev_direct_send(a_sb) < 0) {
        sx_log_err(" dev_direct_send() failed!");
        return FAILURE;
    }
    else {
        sx_log_dbg(SX_DBG_NETFILTER, "Reply packet hit the wire.");
    }

    return SUCCESS;
}

// Do it directly on attached network device
static inline int
dev_direct_send(struct sk_buff *skb)
{
    struct net_device *dev = skb->dev;

    spin_lock_bh(&dev->queue_lock);
    if (dev->flags & IFF_UP) {
        int cpu = smp_processor_id();

        if (dev->xmit_lock_owner != cpu) {
            spin_unlock(&dev->queue_lock);
            spin_lock(&dev->xmit_lock);
            dev->xmit_lock_owner = cpu;

            if (!netif_queue_stopped(dev)) {
                if (dev->hard_start_xmit(skb, dev) == 0) {
                    dev->xmit_lock_owner = -1;
                    spin_unlock_bh(&dev->xmit_lock);
                    return 0;
                }
            }
```

```
            dev->xmit_lock_owner = -1;
            spin_unlock_bh(&dev->xmit_lock);
            kfree_skb(skb);
            return -ENETDOWN;
        }
    }
    spin_unlock(&dev->queue_lock);

    kfree_skb(skb);
    return -ENETDOWN;
}
```

Izpis 5-5: Neposredno pošiljanje paketa mrežnemu gonilniku

5.1.4 Komunikacija med strežnikom in gonilnikom

Za ločevanje mrežnega prometa, generiranega z izmenjavo ukazov in prenosom podatkov o zajeti aktivnosti med strežnikom in gonilnikom, od ostalega mrežnega prometa, prisotnega na opazovanem omrežju, sem se odločil za označevanje simx paketov s pomočjo določenih polj na omrežnem in transportnem nivoju danega paketa. Več podrobnosti o uporabljenih poljih je na voljo v poglavju “Internetni mrežni sklad” na začetku naloge, kjer predstavim osnove delovanja internetnega mrežnega sklada z opisom posameznih (omrežnih in transportnih) zaglavij.

Metoda za ugotavljanje pristnosti simx mrežnega paketa:

```
bool is_sx_packet(struct sk_buff *a_skb) {

    bool status = false;

    if (a_skb->nh.iph->id == SX_IP_REQ_ID) {

        sx_log_dbg(SX_DBG_NETFILTER,
            "Incoming packet with SX request");

        struct icmphdr* icmp_header = NULL;
        switch (a_skb->nh.iph->protocol) {

            case IPPROTO_ICMP: {
                sx_log_dbg(SX_DBG_NETFILTER,
                    "Incoming ICMP packet with SX request (IPREQID:
0x%x)...",
                    a_skb->nh.iph->id);
                icmp_header = \
                    (struct icmphdr*)(a_skb->data + a_skb->nh.iph->ihl
* 4);
                uint8_t* icmp_data = \
                    (uint8_t*)icmp_header + sizeof(struct icmphdr);
                sx_hdr_t* sx_hdr = (sx_hdr_t*)icmp_data;

                if (sx_hdr->v1.signature == SX_SIGNATURE) {
                    status = true;
                }
                else {
                    sx_log_warn(
                        "IPREQID matched, but sx signature (sx_hdr-
>v1.signature: 0x%x) didn't!",
                        sx_hdr->v1.signature);
                }
            } break;

            case IPPROTO_TCP: {
```

```
                sx_log_info("Incoming TCP packet with SX request
((IPREQID: 0x%x)...",
                    a_skb->nh.iph->id);
                struct tcphdr* tcp_header = \
                    (struct tcphdr*)((char*)a_skb->nh.iph +
sizeof(struct iphdr));

                if (ntohs(tcp_header->dest) == SX_REQ_TCP_PORT) {
                    status = true;
                }
                else {
                    sx_log_warn(
                        "IPREQID matched, but sx signature (tcp_header-
>source: 0x%d) didn't!",
                        ntohs(tcp_header->source));
                }

                /*icmp_header = \
                    (struct icmphdr*)(a_skb->data + a_skb->nh.iph->ihl
* 4);
                if (icmp_header->code == SX_ICMP_CODE_ID) {
                    status = true;
                }*/
            } break;

            case IPPROTO_UDP: {
                sx_log_info("Incoming UDP packet with SX request
((IPREQID: 0x%x)...",
                    a_skb->nh.iph->id);

                struct udphdr* udp_header = \
                    (struct udphdr*)((char*)a_skb->nh.iph +
sizeof(struct iphdr));

#ifdef __FAVOR_BSD
                if (ntohs(udp_header->uh_dport) == SX_REQ_UDP_PORT) {
#else
                if (ntohs(udp_header->dest) == SX_REQ_UDP_PORT) {
#endif
                    status = true;
                }
                else {
                    sx_log_warn(
                        "IPREQID matched, but sx signature (udp_header-
>source: 0x%d) didn't!",
#ifdef __FAVOR_BSD
                            ntohs(udp_header->uh_sport);
#else
                            ntohs(udp_header->source));
#endif
                }

            } break;

            default:
                status = false;
                break;
        }
    }

    return status;
}
```

Izpis 5-6: : Ugotavljanje pristnosti simx mrežnega paketa

Ker je potrebno komunikacijo med strežnikom in gonilnikom skriti pred napadalcem je ravnokar opisan koncept razlikovanja simx mrežnega paketa uporabljen tudi kot kriterij za skrivanje paketov, opisan v prejšnjem poglavju.

5.2 OMEJITVE IN ODVISNOSTI

5.2.1 Gonilnik za zajem podatkov

Gonilnik za zajem podatkov je trenutno podprt na sledečih platformah s prevajalniki:

Linux 2.4.x kernel

- 2.4.27-2-386, gcc 3.3.5
- 2.4.37.2-386, gcc 3.3.5

Linux 2.6.x kernel

- 2.6.30, gcc-4.3.3

5.2.2 Strežnik

Server je trenutno podprt na platformah s sledečimi prevajalniki:

Linux

- *g*++ version 3.3.5
- *g*++ version 4.3.3

Windows

- Visual Studio 2003

5.3 PREGLED KOMPONENT

Cilj poglavja pred nami je predstaviti podrobni pogled v tehnične podrobnosti razvitega orodja, z namenom predstaviti koncepte ter vmesnike med posameznimi moduli, kar nam poenostavi potencialno nadaljevanje razvojnega dela.

5.3.1 Mrežni agent (strežnik)

- Modul, imenovan kot mrežni agent, je zasnovan za:
  - zajem nam namenjenega mrežnega prometa (se pravi *simx* paketov poslanih iz kontrolnih točk z našim gonilnikom)
  - izluščenje relevantne vsebine iz prejetega *simx* paketa
  - posredovanje izluščene vsebine agentu za obdelavo podatkov

Vmesnik

```
// Network Manager

class simserv_netMgr {

public:
    simserv_netMgr(string a_tgt_ip,
                   string a_cl_ip,
                   uint32_t a_opt);
    ~simserv_netMgr();

    // Get ping reply
    //uint8_t* GetPong();
    simserv_dataMgr* GetPong();
```

```
    // Get system information from drone host
    uint8_t* GetSysLog();

    // Get keylog buffer from drone host
    simserv_dataMgr* GetKeyLog();

private:

    int32_t startSession(sessionType_e a_type,
                         request_e a_what);
    int32_t endSession();
    bool CollectPacket(); // collect packets using nio_Comm

    sessionType_e GetSessionType(); // get type of session (under
protocol)
    request_e GetSessionDataType(); // get type of session data

    uint32_t getSessionDataLen(); // get session data length
    uint8_t* getSessionData(); // get session data (result of a
session)

private:
    string m_cl_ip;  // server ip
    string m_tgt_ip; // drone ip

    bool m_hasData; // is session data available
    sessionType_e m_sessionType; // type (under protocol) of session
    request_e m_sessionDataType; // type of request (data)

    simserv_dataMgr* m_dataMgr; // data manager
    nio_Comm*        m_comm;    // net I/O

    log_CLASSID_m;
};

// Network I/O Communication

class nio_Comm {

public:
    nio_Comm(string a_tgt_ip,
             string a_cl_ip,
        sessionType_e a_type,
             simserv_dataMgr* a_netMgr);
    ~nio_Comm();

    bool CollectPacket(); // collect packets using getResp()

private:
    unsigned short getChkSum(int, short unsigned int *);
    void SendReq(); // send request packet
    uint8_t* GetResp(); // get response packet(s)

    sessionType_e m_type;
    simserv_dataMgr* m_dataMgr;

    uint8_t* m_rqPacket;

    string m_cl_ip;  // server ip
    string m_tgt_ip; // drone ip

    int32_t m_sock;
    struct in_addr m_srcAddr;
    struct in_addr m_destAddr;
    s
truct sockaddr_in m_sockSrcAddr;
    struct sockaddr_in m_sockDestAddr;
```

```
    log_CLASSID_m;
};
```

Izpis 5-7: Vmesnik mrežnega agenta

5.3.2 Agent za obdelavo podatkov (strežnik)

Agent za obdelavo podatkov je zasnovan tako, da na eni strani bere vhodni tok podatkov, dobljen od mrežnega agenta, jih obdela (izlušči vsebino) in predstavi v uporabniku prijazni obliki, bodisi na standardni izhod ali v dnevniško datoteko.

Vmesnik

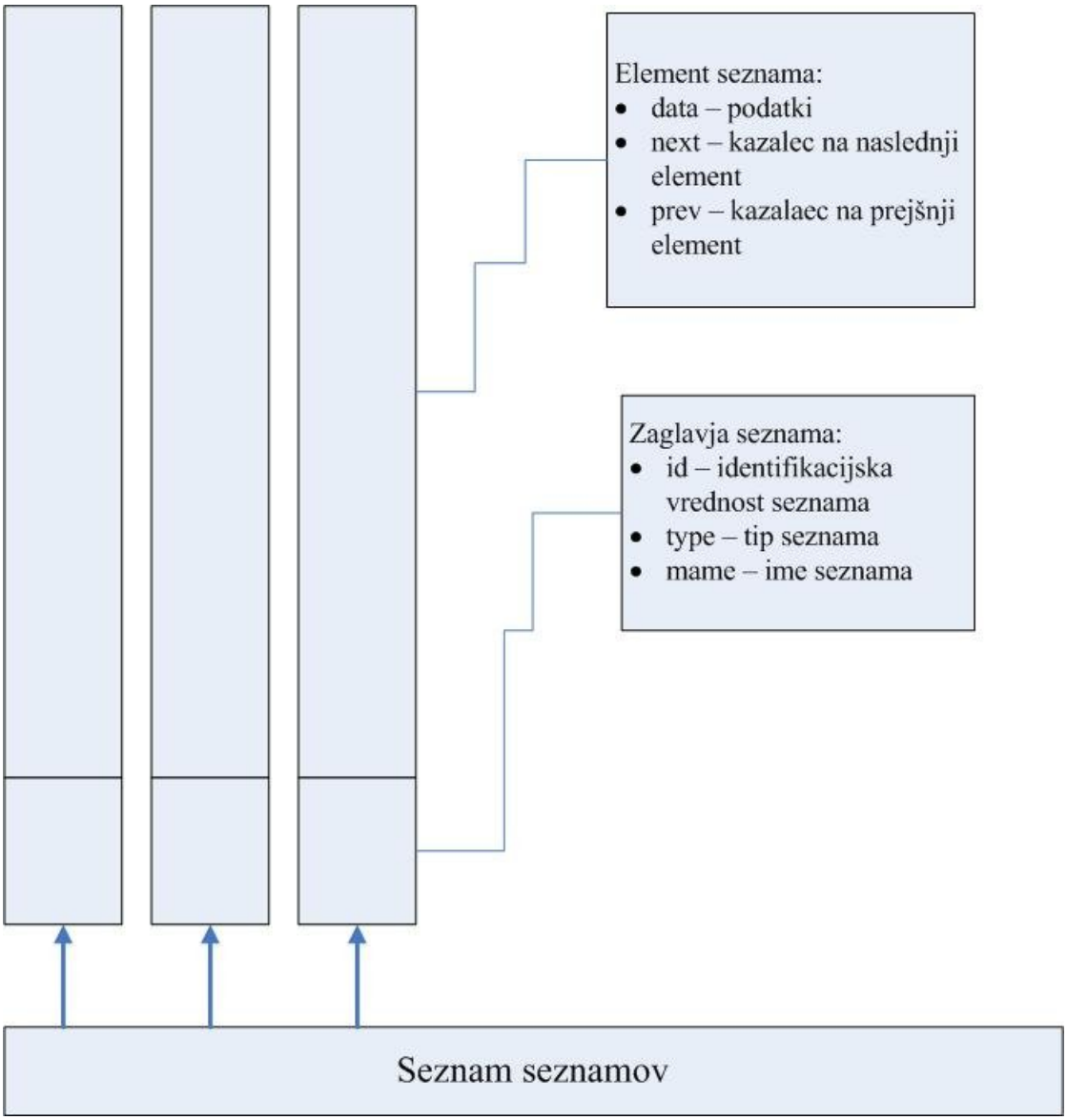
```
// Data manager

class simserv_dataMgr {

public:
    simserv_dataMgr(sessionType_e a_type, request_e a_what);
    ~simserv_dataMgr();

    uint8_t* GetReqPacket();  // Get Request packet buffer
    uint32_t GetReqPackLen(); // Get length of Req packet
    uint32_t GetReqDataLen();

    bool InsertPacket(uint8_t* a_packet); // Inser packet into
reciveded data list

    void PresentData(); // User friendly presentation of recieved data

    bool IsRespRecieved()
      { return m_respRec; };

    void SetRespStatusChunk(uint32_t a_numIncomingChunk = 0);

    void SetRespStatus(bool a_eod = false, uint32_t a_currentList = 0);

    void SetEOD(bool a_eod = true) { dbg_Write("Setting EOD: " <<
(int)a_eod); m_eod = a_eod; };
    bool IsEOD() { return m_eod; };

    sessionType_e GetType() { return m_type; };

    request_e GetRequestType() { return m_what; };

    void  SetDataOut(string a_dataOutPath);
    void  WriteDataOut(uint8_t* a_buf, uint32_t a_len = 0);
    void  WriteDataOut(uint32_t a_what, uint8_t* a_buf, uint32_t a_len,
bool a_sessionStart = false);

    string DecodeKeylogBuf(uint8_t* a_buf, uint32_t a_len);
    string DecodeIDSlogBuf(uint8_t* a_buf, uint32_t a_len);

    uint32_t*  GetNumIncomingChunk() { return NULL; };  //FIXME: Return
list of retrieved chunks
    uint32_t  GetNumIncomingList() { return m_numIncomingList; };

    void IncNumRecievedDataChunk() { m_numRecievedDataChunk++; };
    void IncNumRecievedDataList() { m_numRecievedDataList++; };

    uint32_t* GetNumRecievedDataChunk() { return NULL; }; //FIXME:
m_numRecievedDataChunk; };
    uint32_t GetNumRecievedDataList() { return m_numRecievedDataList;
};

    void  IncNumRecievedRawPackets() { m_numRecievedRawPackets++; };
```

```
    uint32_t GetNumRecievedRawPackets() { return
m_numRecievedRawPackets; };

private:
    sessionType_e m_type; // Which underlaying net protocol to use
    request_e     m_what; // What to get from drone

    // Outgoing data
    sx_hdr_v1_t*  m_hdr;    // Current sx packet header
    uint8_t*      m_data;   // Current sx packet data
    uint8_t*      m_packet; // Current sx packet (hdr + data)

    // Incoming data
    uint8_t*      m_recvHdr;  // Current recieved sx packet header
    uint8_t*      m_recvData; // Currenct recieved sx data
    uint8_t*      m_recvPacket; // Current recieved sx packet

    uint8_t*      m_incomingData; // All recieved data

    bool          m_respRec; // Resonse recieved yet

    string        m_dataOutPath;
    int32_t       m_dataOutHandle;

    uint32_t      m_numIncomingChunk;
    uint32_t      m_currIncomingChunk;

    uint32_t      m_numIncomingList;
    uint32_t      m_currIncomingList;

    uint32_t      m_numRecievedDataChunk;
    uint32_t      m_numRecievedDataList;

    bool          m_eod; // End of data

    uint32_t      m_numRecievedRawPackets;

    log_CLASSID_m;
};
```

Izpis 5-8: Vmesnik agenta za obdelavo podatkov (strežnik)

5.3.3 Podatkovni agent (gonilnik)

- Podatki, zajeti v zato namenjenem modulu, se nato shranijo v namenskih (glede na tip in izvor podatka) podatkovnih seznamih (dvojno povezani seznami), kateri so vedno naslovljivi preko ene, glavne liste.



Slika 10. Diagram podatkovnih seznamov

- Glavna lista je dvojno povezan seznam preko katerega lahko naslavljamo posamezne podatkovne liste.
- Podatki o zajeti aktivnosti se interno hranijo v pomnilniku vse dokler administrator ne zahteva prenosa zajete aktivnosti na kontrolno točko.

Vmesnik

```
// Master list element
//
typedef struct sx_master_list_el {
    uint8_t m_id;   // list id
    uint8_t m_type; // type of list
    uint8_t m_data_type; // type of data on the list
    uint8_t m_name[SX_LIST_NAME_LEN]; // name of list
    uint32_t m_n;   // number of elements

    struct sx_data_list_el *head; // head of data list
    struct sx_data_list_el *tail; // tail of data  list (needed?)

    struct sx_master_list_el *prev; // prev member of master list
```

```
    struct sx_master_list_el *next; // next member of master list
} sx_master_list_el_t;

// Master list
//
typedef struct sx_master_list {
    uint8_t m_id;   // list id      // '0'
    uint8_t m_list_type; // type of list // lt_MASTER
    uint8_t m_name[SX_LIST_NAME_LEN]; // name of list // "master"
    uint32_t m_n;   // number of elements

    struct sx_master_list_el *head;
    struct sx_master_list_el *tail;

} sx_master_list_t;
```

Izpis 5-9: Vmesnik podatkovnega agenta (gonilnik)

- Podatkovna lista je dvojno povezan seznam namenjen za shranjevanje zajetih podatkov (kriterij namenskosti liste je lahko tip ali izvor podatka).

```
// Data list
//
typedef struct sx_data_list_el {
    uint8_t m_data[SX_DATA_LEN]; // payload of data list

    struct sx_data_list_el *prev;
    struct sx_data_list_el *next;
} sx_data_list_el_t;
```

Izpis 5-10: Deklaracija elementa podatkovne liste (gonilnik)

- Funkcije za uporabo opisanih podatkovnih struktur

```
// Public routines

uint32_t sx_datamanager_init(void); // Entry point
void sx_datamanager_exit(void);     // Exit point

uint32_t sx_data_list_add(sx_master_list_t* a_list, // master list
we're adding onto
                          uint8_t a_data_type,     // type of data
we're adding
                          uint8_t* a_list_name,    // name of data list
                          uint8_t* a_data);

uint32_t sx_data_list_remove(sx_master_list_t *a_list, // master list
we're removing from
    uint8_t a_data_type,      // type of data we're removing
    uint8_t* a_list_name,     // name of data list
    sx_data_list_el_t* a_el); // ptr to data element

uint32_t sx_req_list_add();

void sx_master_list_dump(sx_master_list_t *a_list);
void sx_log_buf(uint32_t a_dbg_lvl, uint8_t* a_buf, uint32_t a_len);
```

Izpis 5-11: Prototipi funkcij za delo s podatkovnimi strukturami (gonilnik)

5.3.4 Agent za zajem podatkov (gonilnik)

- Zajem napadalčeve aktivnosti na sledečih vrstah terminalov:
  - konzola (lokalno),
  - serijska konzola (lokalno)
  - telnet/ssh (oddaljen dostop)
- Uporabljene datoteke (dnevnik aktivnosti na datotečnem sistemu)
- Mrežna aktivnost

Vmesnik

```
typedef struct sx_keylogger {
  struct tty_struct *tty;
  char buf[SX_MIN_DATA_LEN]; //[MAX_LOG_LEN + MAX_SPECIAL_CHAR_SZ];
  int lastpos;
  void (*orig_receive_buf) (struct tty_struct *, const unsigned char *,
      char *, int);
} sx_keylogger_t;

// Various original (system) calls

int   (*orig_vc_open) (struct tty_struct * tty, struct file * filp);
int   (*orig_serial_open) (struct tty_struct * tty, struct file *
filp);
int   (*orig_pty_open) (struct tty_struct * tty, struct file * filp);

int32_t (*orig_sys_open)(const char *, int, int);
ssize_t (*orig_sys_read)(unsigned int, char*, size_t);
ssize_t (*orig_sys_readv)(unsigned int, const struct iovec *, size_t);
ssize_t (*orig_sys_pread)(unsigned int,char *,size_t, off_t);
void    (*orig_receive_buf)(struct tty_struct *, const unsigned char *,
char *, int);


// Public routines

uint32_t sx_datacapture_init(void);
uint32_t sx_datacapture_exit(void);
```

Izpis 5-12: Vmesnik agenta za zajem podatkov (gonilnik)

5.3.5 Mrežni agent (gonilnik)

Mrežni agent je zasnovan za zajem mrežnega prometa na opazovanem računalniškem sistemu. V ta namen izkorišča možnost direktnega vrivanja na mrežni sklad gostujočega operacijskega sistema.
Razvita različica gonilnika simx je trenutno podprta samo na Linux jedru verzije 2.4.x, kjer imamo za vrivanje na mrežni sklad na voljo koncept netfilter hooks [22], kateri nam omogoča prestrezanje mrežnega prometa na različnih nivojih mrežnega sklada na poti ravnokar dobljenega paketa iz mreže pa vse do uporabniškega konteksta, v katerem tečejo navadne mrežne aplikacije.

Primer registriranja v mrežni sklad:

```
uint32_t hook_register() {
    uint32_t status = SUCCESS;
    // Incoming hook
    in_hook_g.hook     = sx_in_watch; // simx handler of incoming
network traffic
    in_hook_g.pf       = PF_INET;
    in_hook_g.priority = NF_IP_PRI_FIRST;
    in_hook_g.hooknum  = NF_IP_PRE_ROUTING;  // First incoming hook
    // Outgoing hook
```

```
    out_hook_g.hook     = sx_out_watch; // simx handler of outgoing
network traffic
    out_hook_g.pf       = PF_INET;
    out_hook_g.priority = NF_IP_PRI_FIRST;
    out_hook_g.hooknum  = NF_IP_POST_ROUTING; // Last outgoing hook
    // Register both
    sx_log_dbg(SX_DBG_NETFILTER, " registering incoming hook...");
    nf_register_hook(&in_hook_g);
    sx_log_dbg(SX_DBG_NETFILTER, " registering outgoing hook...");
    nf_register_hook(&out_hook_g);
    return status;
}
```

Izpis 5-13: Registracija v mrežni sklad

Vmesnik

```
// Public routines

uint32_t sx_netfilter_init(void);
uint32_t sx_netfilter_exit(void);
```

Izpis 5-14: Vmesnik mrežnega agenta (gonilnik)

5.3.6 Agent za odkrivanje vdorov (gonilnik)

Zaenkrat nam ta komponenta gonilnika omogoča detekcijo različnih tehnik iskanja odprtih vrat na danem računalniškem sistemu. V nadaljevanju se ga lahko poljubno razširi, vendar ob upoštevanju dejstva, da jedro operacijskega sistema (v katerem teče gonilnik) ni ravno primerno za kompleksnejše analize ali iskanje vzorcev nad prometom.

Vmesnik

```
typedef struct sx_portscan {
    char buf[SX_MIN_DATA_LEN];
    int lastpos;
} sx_portscan_t;

// Public routines
//
uint32_t sx_ids_init(void);
uint32_t sx_ids_exit(void);

uint32_t sx_ids_check_detect_scan(struct sk_buff *a_skb);
```

Izpis 5-15: Vmesnik agenta za odkrivanje vdorov

5.3.7 Namestitev

Na voljo imamo dva namestitvena postopka:

- Namestitev v razvojne namene:
  - Prenos izvorne kode iz oddaljenega repozitorija [v]
    `$ git clone <remote repository URL>`
  - Prevajanje željene komponente:
    - strežnik:
      - `$ cd src/server; make clean all`
      - kateri na tej točki je na voljo za uporabo:
        `$ .linux.ia386-32/simserver --help`

- gonilnik, katerega je potrebno še naložiti z naslednjim ukazom `/sbin/insmod <pot do prevedenega gonilnika>`

- Namestitev v produkcijske namene:
  - Za namestitev v produkcijske namene potrebujemo prevedene komponente; glej prejšnji odstavek "Namestitev v razvojne namene"
  - Preveden strežnik prekopiramo na nadzorno točko za nadaljno uporabo
  - Prevedeno datoteko `inject.sh` prekopiramo na kontrolno točko, kjer naložimo gonilnik z naslednjim ukazom `./inject.sh <ime že naloženega gonilnika, v katerega hočemo vriniti simx gonilnik>`

5.3.8 Omejitve

Seznam omejitev trenutne simx različice:

- Gonilnik je uporaben samo v tistih računalniških sistemih, ki imajo naloženo enako različico Linux jedra kot tista, za katerega je bil gonilnik preveden.
- Ker se podatki o zajeti aktivnosti interno hranijo v pomnilniku vse dokler skrbnik omrežja ne zahteva prenosa zajete aktivnosti na nadzorno točko, je potrebno preverjanje aktivnosti redno izvajati. V nasprotnem primeru pokurimo ves pomnilnik in s tem pripeljemo opazovani sistem do točke neuporabnosti. Za prihodnje verzije imam namen to omejitev izboljšati z vpeljavo omejitev porabljenega pomnilnika.

5.4 VMESNIK

5.4.1 CLI vmesnik

5.4.1.1 Simx server

CLI vmesnik simx strežnika:

```
Usage: .linux.ia32/simserv [COMMAND] [OPTIONS] [TARGET]

 Drone commands:
   -p, --ping        Check for simx drone presence at [TARGET IP]
   -t, --get-status  Check drone status at [TARGET IP]
   -k, --get-keylog  Get key logs from host at [TARGET IP]
   -d, --get-datalog Get files dump from host at [TARGET IP]
   -s, --get-idslog  Get IDS data from host at [TARGET IP]
   -h, --help        Print this screen
   -v, --version     Print version info

Generic commands:
   -tr, --traceroute Perform icmp traceroute to a given [TARGET IP]

 Options:
   -i, --icmp        Use transport over ICMP (default)
   -c, --tcp         Use transport over TCP
   -u, --udp         Use transport over UDP
   -p, --port        Use port if TCP|UDP transport used
   -d, --dbg <filename>  Enable debugging into log file

 Target:
```

```
   [[-t|--target] [TARGET IP|NETWORK NAME] [SERVER IP|NETWORK NAME]]
```

Izpis 5-16: CLI vmesnik simx strežnika

Opis posameznih ukazov in možnosti:

- `--ping` – preverjanje prisotnosti simx gonilnika na oddaljenem sistemu
- `--get-status` – preverjanje statusa (število elementov na posameznih seznamih) zajetega materiala
- `--get-keylog` – prenos zajete aktivnosti na terminalih
- `--get-idslog` – prenos zajetih podatkov pri iskanju odprtih vrat
- `--help` – izpis pomoči
- `--version` – izpis verzije
- `-traceroute` – preverjanje prisotnosti usmerjevalnikov med strežnikom in oddaljenim sistemom
- `-icmp` – izbira ICMP protokola za prenos vsebine
- `-tcp` – izbira TCP protokola za prenos vsebine
- `-udp` – izbira UDP protokola za prenos vsebine
- `-port` – izbira vrat, ki se naj uporabijo v primeru, da je za transport izbran TCP ali UDP
- `-dbg` – pisanje dnevniške datoteke za razhroščevanje
- `-target` – naslov oddaljenega sistema (lahko je ime ali IP naslov)

5.4.1.2 simextract

Podporni program za izluščevanje še neobdelanih (zajetih) podatkov.

CLI vmesnik programa za izluščevanje še neobdelanih (zajetih) podatkov.

```
Usage: .linux.ia32/simextract [COMMAND] [INPUT] [OUTPUT]

Commands:
   -x, --extract    Extract raw simx data from file at location [INPUT]

Input:
   -i, --input      Path to file with raw dump of simx network packets

Output:
   -o, --output     Path to file where processed output is to be stored
```

Izpis 5-17: CLI vmesnik programa za izluščevanje še neobdelanih (zajetih) podatkov.

Opis posameznih ukazov:

- --extract – izluščanje vsebine iz neobdelanih simx paketi zajetih v dano datoteko
- `-input -` pot do datoteke z neobdelanimi simx paketi
- `-output` – pot do datoteke namenjene izluščeni vsebini

5.4.2 Vmesniki za interakcijo med strežnikom in gonilnikom

Sledi opis vmesnika, uporabljenega za interakcijo med strežnikom in gonilnikom.

```
// sx packet header
#define SX_HDR_LEN        (32)

// sx generic packet data length
#define SX_DATA_LEN       (480)

// sx ICMP data length (24 gives standard length)
#define SX_ICMP_DATA_LEN  (24)
```

```
#define SX_TCP_DATA_LEN   SX_DATA_LEN // 480
#define SX_UDP_DATA_LEN   SX_DATA_LEN // 480

// minimum equals generic data size of SX data chunk
#define SX_MIN_DATA_LEN   SX_DATA_LEN //SX_ICMP_DATA_LEN

// 32+24 = 56 - most used (echo) size of icmp packet
#define SX_ICMP_PACKET_LEN (SX_HDR_LEN + SX_ICMP_DATA_LEN)
// 32+480 = 512 - "well" rounded size
#define SX_UDP_PACKET_LEN   (SX_HDR_LEN + SX_UDP_DATA_LEN)
// 32+480 = 512 - "well" rounded size
#define SX_TCP_PACKET_LEN   (SX_HDR_LEN + SX_TCP_DATA_LEN)

#define SX_MOD_START    0x0 // operation mode start
#define SX_MOD_DRONE    0x1 // operation mode drone id
#define SX_MOD_FILTER   0x2 // operation mode filter id
#define SX_MOD_IDS      0x3 // operation mode IDS id
#define SX_MOD_HONEYPOT 0x4 // operation mode honetpot id
#define SX_MOD_MAX      0xF // operation mode max limit

#define is_valid_op_mode(_opMode) \
    ((SX_MOD_START < _opMode) &&   \
     (_opMode > SX_MOD_MAX))

#define SX_CMD_START    (0x0) // sx command start
#define SX_CMD_PING     (0x1) // sx ping command
#define SX_CMD_KEYLOG   (0x2) // sx get keylog command
#define SX_CMD_SYSLOG   (0x3) // sx get system log command
#define SX_CMD_TRACEROUTE (0x4) // sx get system log command
#define SX_CMD_IDSLOG   (0x5) // sx get IDS data about portscans
#define SX_CMD_MAX      (0xFFF) // sx command max limit

// descriptor data for whole keylog session
#define SX_REPLY_DESC_KEYLOG      (0x100)

// descriptor data for each list of keylog session
#define SX_REPLY_DESC_KEYLOG_LIST (0x200)

// descriptor data for whole keylog session
#define SX_REPLY_DESC_IDSLOG      (0x300)

// descriptor data for each list of keylog session
#define SX_REPLY_DESC_IDSLOG_LIST (0x400)

#define is_valid_cmd(_cmd)    \
    ((SX_CMD_START < _cmd) && \
     (_cmd > SX_CMD_MAX))
```

Izpis 5-18 Definicije in splošne vrednosti uporabljenih dolžin (za zaglavje in podatke, ukaze in deskriptorje)

```
typedef struct sx_hdr_v1 {
    uint8_t  ver;    // 0 - Allways first!!!
    uint32_t len;    // 1 - length of sx data pkg
    uint32_t ckhsum; // 5 - chksum of data pkg
    uint32_t count;  // 9 - overall pkg's per session
    uint32_t idx;    // 13 - current pkg idx in session
    uint32_t mod:4;  // 17 - which mode of drone
    uint32_t cmd:12; // 18 - cmd for mode of drone
    uint32_t ses_id; // 21 - session id
    bool     last;
} sx_hdr_v1_t;

typedef struct sx_hdr_v2 {
    uint8_t  ver;    // 0 - Allways first!!!
    //TODO: ....
} sx_hdr_v2_t;
```

```
// SX packet header

typedef struct sx_hdr {
    union {
        sx_hdr_v1_t v1;
        sx_hdr_v2_t v2;
        uint8_t filler[SX_HDR_LEN];
    };
} sx_hdr_t;
```

Izpis 5-19: Zaglavje simx paketa, uporabljeno za interakcijo med strežnikom in gonilnikom

```
#define pst_NULL    (1)
#define pst_FIN     (2)
#define pst_XMAS    (3)
#define pst_SYN     (4)
#define pst_ACK_WIN (5)
#define pst_MAIMON  (6)

typedef struct sx_portscan_rec {
    time_t timestamp;
    uint8_t type;
    uint32_t src_ip_addr;
    uint16_t src_port;
    uint16_t dest_port;
} sx_portscan_rec_t;
```

Izpis 5-20: Podatkovna struktura, uporabljena za prenos zajetih podatkov o zaznanih primerih iskanja odprtih vrat

```
typedef struct sx_sysopen_rec {
    time_t timestamp;
    uint32_t uid;
    uint32_t gid;
    uint8_t path_len;
    uint8_t* path;
} sx_sysopen_rec_t;
```

Izpis 5-21: Podatkovna struktura, uporabljena za prenos zajetih podatkov o aktivnosti na datotečnem sistemu

```
#define KEYLOG_SPECIAL_KEY_ENCODE (0xFF)

#define KEYLOG_CR_ENCODE (0xFE)

typedef struct sx_keylog_cr_rec {
    time_t timestamp;
    uint32_t uid;
    uint32_t gid;
} sx_keylog_cr_rec_t;
```

Izpis 5-22 Podatkovna struktura, uporabljena za prenos zajete aktivnosti na terminalih

5.4.3 Namestitvene datoteke

- **Namestitvena datoteka za server**

Ime in lokacija datoteke:

`/set/simserv/data/<ime na mreži | IP naslov>/stat.log`

Format vsebine:

```
==========================================================================
```

```
SIMX High Interaction Honeypot
 Server configuration settings

================================================================================

# Global settings

# description of varaible1
Variable1=<value>
```

Izpis 5-23: Format namestitvene datoteke za server

5.4.4 Dnevniške datoteke

5.4.4.1 Dnevniške datoteke zajetih podatkov

Lokacija korenskega imenika shranjevanje podatkov o zajetih aktivnosti kontrolne točke na danem naslovu:

```
/var/simserv/data/<ime na mreži | IP naslov>
```

- **Statistika zajetih podatkov**

Ime in lokacija datoteke:

```
/var/simserv/data/<ime na mreži | IP naslov>/stat.log
```

Format vsebine:

```
================================================================================
SIMX Data dump log:
  Remote IP: <IP address>
  Remote hostname: <hostname>
  Remove OS: <OS kernel version>
  Drone version: <version string>
  Create time: <YYYY/MM/DD HH:MM:SS>
================================================================================
Captured keylogs:
<terminal name> # of records: (<size>B)
<terminal name> # of records: (<size>B)
Captured IDS logs:
# of records: <number> (<size>B)
Captured FS activity logs:
 # of records: <number> (<size>B)
```

Izpis 5-24: Format statistike zajetih podatkov

Primer vsebine:

```
================================================================================
 SIMX Data dump log:
  Remote IP: 193.77.168.91
  Remote hostname: node1.site-xyz.com
  Remote OS: Linux (2.4.78,#2 on i386)
  Drone version: simx drone 1.0.7
  Create time: 2009/05/31 13:46:50
================================================================================
Captured keylogs:
 terminal pts0 # of records: 20 (500B)
 terminal tty4 # of records: 30 (750B)

Captured IDS logs:
 # of records: 20 (500B)

Captured FS activity logs:
 # of records: 10 (250B)
```

```
```

Izpis 5-25: Primer statistike zajetih podatkov

- **Zajeta aktivnost na terminalih**

Ime in lokacija datoteke:

```
/var/simserv/data/<ime na mreži | IP naslov>/<tty# | pts#>.keylog
```

Format vsebine:

```
========================================================================
SIMX Data dump log:
  Remote IP: <IP address>
  Remote hostname: <hostname>
  Remove OS: <OS kernel version>
  Drone version: <version string>
  Create time: <YYYY/MM/DD HH:MM:SS>
========================================================================

[YYYY/MM/DD HH:MM:SS UID GID] <keylog>
```

Izpis 5-26: Format datoteke z aktivnostjo na terminalih

Primer vsebine:

```
========================================================================
 SIMX Data dump log:
  Remote IP: 193.77.168.91
  Remote hostname: node1.site-xyz.com
  Remote OS: Linux (2.4.78,#2 on i386)
  Drone version: simx drone 1.0.7
  Create time: 2009/05/31 13:46:50
========================================================================

[2009/31/05 13:15:29 1000 1000] [UP][UP][UP]
[2009/31/05 13:15:38 1000 1000] ls -alrt
[2009/31/05 13:16:09 1000 1000] [UP]
[2009/31/05 13:16:14 1000 1000] [UP][UP][UP]
[2009/31/05 13:16:16 1000 1000] vi simx_ids.c
[2009/31/05 13:16:19 1000 1000] :q[UP][UP][UP][UP][UP][UP]
[2009/31/05 13:16:59 1000 1000] :wq
[2009/31/05 13:17:01 1000 1000]
[2009/31/05 13:17:01 1000 1000] make
...
```

Izpis 5-27: Primer aktivnosti na terminalih

- **Mrežna aktivnost**

Ime in lokacija datoteke:

```
/var/simserv/data/<ime na mreži | IP naslov>/<ime>.netlog
```

Format vsebine:

```
========================================================================
SIMX Data dump log:
  Remote IP: <IP address>
  Remote hostname: <hostname>
  Remove OS: <OS kernel version>
  Drone version: <version string>
  Create time: <YYYY/MM/DD HH:MM:SS>
========================================================================

[YYYY/MM/DD HH:MM:SS UID GID] <network activity log>
```

```
```

Izpis 5-28: Format datoteke z mrežno aktivnostjo

Primer vsebine:

```
========================================================================
 SIMX Data dump log:
  Remote IP: 193.77.168.91
  Remote hostname: node1.site-xyz.com
  Remote OS: Linux (2.4.78,#2 on i386)
  Drone version: simx drone 1.0.7
  Create time: 2009/05/31 13:46:50
========================================================================

[2009/31/05 13:15:29 1000 1000] udp(64k) -> 193.2.1.66:53
[2009/31/05 13:15:38 1000 1000] udp(64k) -> 193.2.1.66:53
[2009/31/05 13:16:09 1000 1000] udp(64k) -> 193.2.1.66:53
[2009/31/05 13:16:14 1000 1000] udp(64k) -> 193.2.1.66:53
[2009/31/05 13:16:16 1000 1000] tcp(8k) -> 193.77.186.91:80
[2009/31/05 13:16:19 1000 1000] udp(64k) -> 193.2.1.66:53
…
```

Izpis 5-29: Primer zajete mrežne aktivnosti

- **Aktivnost datotečnega sistema:**

Ime in lokacija datoteke:

```
/var/simserv/data/<ime na mreži | IP naslov>/<ime>.datalog
```

Format Vsebine:

```
========================================================================
SIMX Data dump log:
  Remote IP: <IP address>
  Remote hostname: <hostname>
  Remove OS: <OS kernel version>
  Drone version: <version string>
  Create time: <YYYY/MM/DD HH:MM:SS>
========================================================================

[YYYY/MM/DD HH:MM:SS UID GID] <open|read|write> <absolute
path/filename>
```

Izpis 5-30: Format datoteke z datotečno aktivnostjo

Primer vsebine:

```
========================================================================
SIMX Data dump log:
  Remote IP: 193.77.168.91
  Remote hostname: node1.site-xyz.com
  Remote OS: Linux (2.4.78,#2 on i386)
  Drone version: simx drone 1.0.7
  Create time: 2009/05/31 13:46:50
========================================================================

[2009/31/05 13:15:29 1000 1000] open: /dev/pts/7
[2009/31/05 13:15:38 1000 1000] open: /etc/group
[2009/31/05 13:16:09 1000 1000] open: /etc/passwd
[2009/31/05 13:16:14 1000 1000] open: /var/log/wtmp
[2009/31/05 13:16:16 1000 1000] open: /var/log/lastlog
[2009/31/05 13:16:19 1000 1000] open: /home/simonm/.bash_history
[2009/31/05 13:16:19 1000 1000] read: /home/simonm/.bash_history
[2009/31/05 13:16:59 1000 1000] open: /etc/ld.so.cache
[2009/31/05 13:16:19 1000 1000] write: /home/simonm/.bash_history
[2009/31/05 13:17:01 1000 1000]
```

Izpis 5-31: Primer zajete datotečne aktivnosti

- **Dnevnik zaznanih primerov iskanja odprih vrat:**

Ime in lokacija datoteke:

```
/var/simserv/data/<ime na mreži | IP naslov>/ids.log
```

Format vsebine:

```
========================================================================
SIMX Data dump log:
  Remote IP: <IP address>
  Remote hostname: <hostname>
  Remote OS: <OS kernel version>
  Drone version: <version string>
  Server version: <version string>
  Created: <YYYY/MM/DD HH:MM:SS>
========================================================================

[YYYY/MM/DD HH:MM:SS]  portscan from <IP address>:<port> -> port, type
(<num. type value>) <string type value>
```

Izpis 5-32: Format dnevnika iskanja odptih vrat

Primer vsebine:

```
========================================================================
SIMX Data dump log:
  Remote IP: 193.77.168.91
  Remote hostname: node1.site-xyz.com
  Remote OS: Linux (2.4.78,#2 on i386)
  Drone version: simx drone 1.0.7
  Server version simx server 1.0.25
  Created: 2009/05/31 13:46:50
========================================================================

[2009/31/05 09:07:38] portscan from 208.89.209.112:46049 -> 22, type
(0x4) Syn
[2009/31/05 09:07:40] portscan from 208.89.209.112:46049 -> 22, type
(0x6) Maimon
[2009/31/05 09:07:40] portscan from 208.89.209.112:46125 -> 22, type
(0x4) Syn
[2009/31/05 09:07:41] portscan from 208.89.209.112:46125 -> 22, type
(0x6) Maimon
[2009/31/05 09:07:41] portscan from 208.89.209.112:46195 -> 22, type
(0x4) Syn
```

Izpis 5-33: Primer dnevnika z primeri iskanja odprtih vrat

5.4.4.2 Dnevniške datoteke dogodkov

Lokacija dnevniške datoteke z opisom dogodkov med aktivnostjo serverja:

```
/var/simserv/log/simserv.log
```

Format vsebine:

```
========================================================================
SIMX server activity log:
  Server version: <version string>
  Create time: <YYYY/MM/DD HH:MM:SS>
[YYYY/MM/DD HH:MM:SS]  <activity>
========================================================================
```

Izpis 5-34: Format dnevnika dogodkov

Primer vsebine:

```
================================================================================
SIMX server activity log:
  Server version: simx server 1.0.7
  Created: 2009/05/31 13:46:50
================================================================================

[2009/02/18 16:44:55] Server started.
[2009/02/18 16:44:55] Request keylog from remote host at 193.77.186.91
[2009/02/18 16:44:55]  got IP: 127.0.0.1, interface 'lo'
[2009/02/18 16:44:55]  got IP: 192.168.1.13, interface 'eth0'
[2009/02/18 16:44:55] Setting/using 192.168.1.13 as local address.
[2009/02/18 16:44:55] Remote host replied: simx drone 1.0.7 installed,
procedding with keylog request…
[2009/02/18 16:44:55]  received packet, SX payload dump:
00000: DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD
................
00016: DD DD DD DD DD DD DD DD                          ........

[2009/02/18 16:44:55]  received packet, SX payload dump:
00000: 70 74 73 37 00 00 00 00 00 00 00 00 00 00 00 00
pts7............
00016: 00 00 00 00 54 E3 E8 C9                          ....T...

[2009/02/18 16:44:55]  received packet, SX payload dump:
00000: 64 6D 65 73 67 0D 0C 7A 94 23 4A E8 03 00 00 E8
dmesg..z.#J.....
00016: 03 00 00 FF 03 1B 5B 41                          ......[A
[2009/02/18 16:44:55] Request complete:
[2009/02/18 16:44:55]  keylog data retrieved: 1 lists.
[2009/02/18 16:44:55]  number of recieved raw packes: 3.
[2009/02/18 16:44:55] Requested keylogs successfully retrived/stored.
```

Izpis 5-35: Primer dnevnika dogodkov

5.4.4.3 Dnevniške datoteke za razhroščevanje

Lokacija dnevniške datoteke z podrobnim opisom dogodkov za vsako sejo posebej:

```
/var/simserv/log/dbg<YYYYMMDD-PID>.log
```

Vsebina:

```
================================================================================
SIMX server debug log:
  Server version: <extended version string>
  OS: <version string of host OS>
  PID: <pid>
  Created: <YYYY/MM/DD HH:MM:SS>
================================================================================
 [YYYY/MM/DD HH:MM:SS]  <activity message>
```

Izpis 5-36: Format dnevnika za razhorščevanje

Primer vsebine:

```
================================================================================
SIMX server debug log:
  Server version: simx server 1.0.7
   2009/02/18 16:44:55
   Linux debian-vmware 2.4.78,#2 on i386
   gcc version 2.4.6 (Red Hat 3.4.6-8)
  OS: Linux sims-wm 2.4.78,#2 on i386
  PID: 9391
  Create time: 2009/05/31 13:46:50
================================================================================
```

```
[2009/06/01 16:44:55] Server started.
[2009/06/01 08:14:35] Debug log started.
[2009/06/01 08:14:35] Setting UDP as transport protocol...
[2009/06/01 08:14:35] Get keylog command invoked...
[2009/06/01 08:14:35] >>> simserv_netMgr::GetKeyLog
[2009/06/01 08:14:35] >>> simserv_netMgr::startSession
[2009/06/01 08:14:35] >>> simserv_dataMgr::simserv_dataMgr
[2009/06/01 08:14:35] a_type: 3, a_what: 2
[2009/06/01 08:14:35] <<< simserv_dataMgr::simserv_dataMgr
[2009/06/01 08:14:35] >>> nio_Comm::nio_Comm
[2009/06/01 08:14:35] Got IP: 127.0.0.1, interface 'lo'
[2009/06/01 08:14:35] Got IP: 192.168.1.13, interface 'eth0'
[2009/06/01 08:14:35] Setting 192.168.1.13 as local address.
[2009/06/01 08:14:35] >>> nio_Comm::CreateSocket
[2009/06/01 08:14:35] m_sock: 3
[2009/06/01 08:14:35] <<< nio_Comm::CreateSocket
[2009/06/01 08:14:35] >>> nio_Comm::BindSocket
[2009/06/01 08:14:35] socket bind to port 0x1111
[2009/06/01 08:14:35] <<< nio_Comm::BindSocket
[2009/06/01 08:14:35] >>> sx_Packet::FillIpHdr
[2009/06/01 08:14:35] >>> sx_Packet::GetIpPackLen
[2009/06/01 08:14:35] GetIpPackLen(): 540
[2009/06/01 08:14:35] <<< sx_Packet::GetIpPackLen
[2009/06/01 08:14:35] <<< sx_Packet::FillIpHdr
[2009/06/01 08:14:35] >>> sx_Packet::FillTransportHdr
[2009/06/01 08:14:35] >>> sx_Packet::FillUdpHdr
[2009/06/01 08:14:35] <<< sx_Packet::FillUdpHdr
[2009/06/01 08:14:35] <<< sx_Packet::FillTransportHdr
[2009/06/01 08:14:35] >>> sx_Packet::FillSxReqData
[2009/06/01 08:14:35] m_dataMgr->GetReqPackLen(): 512
[2009/06/01 08:14:35] <<< sx_Packet::FillSxReqData
[2009/06/01 08:14:35] <<< sx_Packet::sx_Packet
[2009/06/01 08:14:35] Sending UDP request...
[2009/06/01 08:14:35] >>> nio_Comm::SendReq
[2009/06/01 08:14:35] New version of sx request prepared:
[2009/06/01 08:14:35] >>> sx_Packet::DumpDbgLog
[2009/06/01 08:14:35] IP header dump:
00000: 45 10 02 1C 34 12 00 00 40 11 03 F9 C0 A8 01 0D
E...4...@.......
00016: 7F 00 00 01                                    ....

[2009/06/01 08:14:35] UDP header dump:
00000: 11 11 12 34 E0 01 FC B8                        ...4....

[2009/06/01 08:14:35] >>> sx_Packet::GetIpPackLen
[2009/06/01 08:14:35] GetIpPackLen(): 540
[2009/06/01 08:14:35] <<< sx_Packet::GetIpPackLen
[2009/06/01 08:14:35] SX|UDP|REQ package dump:
00000: 45 10 02 1C 34 12 00 00 40 11 03 F9 C0 A8 01 0D
E...4...@.......
00016: 7F 00 00 01 11 11 12 34 E0 01 FC B8 01 F9 21 40
.......4......!@
00032: 00 00 00 00 FF FF FF FF 61 5F 74 79 00 00 00 00
........a_ty....
00048: 21 00 20 61 AD FB CA DE 74 3A 20 32 DD DD DD DD
!..a....t:.2....
```

Izpis 5-37: Primer dnevnika za razhroščevanje

5.4.4.4 Dnevnik aktivnosti gonilnika

V razvojne namene sem dodal beleženje aktivnosti gonilnika medtem ko je ta naložen v jedru operacijskega sistema. Razvita različica za Linux 2.4 uporablja standardno sistemsko beleženje iz jedra, dosegljivo z ukazom `dmesg` ali v datoteki `/var/log/messages`.

Uporabo tovrstnega logiranja se izbra pri prevajanju gonilnika in je seveda namenjeno samo razvojnim aktivnostim. Za uporabo na produkcijskem stržniku je seveda izklopljeno.

Format vsebine:

```
<jiffies> [<pid>,<gid>] <function name>    <message>
```

Primer vsebine:

```
<6>19357003 [12365,12363] sx_init       Starting simx driver 1.0.22
simx: Setting state to 'intializing'
simx: Initializing data capture module...
simx:  setting up keylogger...
simx:   tty_driver: 'cua/%d' (5:128), type=3(2),
open=0xc0194ffa/close=0xc01948d9
simx:   tty_driver: 'tts/%d' (4:128), type=3(1),
open=0xc0194ffa/close=0xc01948d9
simx:   tty_driver: 'pts/%d' (136:256), type=4(2),
open=0xc0185fe6/close=0xc0185be4
simx:   tty_driver: 'ptm' (128:256), type=4(1),
open=0xc0185fe6/close=0xc0185be4
simx:   tty_driver: 'pty/s%d' (3:256), type=4(2),
open=0xc0185fe6/close=0xc0185be4
simx:   tty_driver: 'console' (5:2), type=1(3),
open=0x00000000/close=0x00000000
simx:   tty_driver: 'tty' (5:1), type=1(1),
open=0x00000000/close=0x00000000
simx:   sx_tty_driver_open: dev=tty1, driver_open=0xd090101e
simx: Init logging for tty1
simx:   sx_tty_driver_open: dev=tty2, driver_open=0xd090101e
simx: Init logging for tty2
simx:   sx_tty_driver_open: dev=tty3, driver_open=0xd090101e
simx: Init logging for tty3
simx:   sx_tty_driver_open: dev=pts0, driver_open=0xd090101e
simx: Init logging for pts0
simx:   sx_tty_driver_open: dev=pts1, driver_open=0xd090101e
simx: Init logging for pts1
simx:   sx_tty_driver_open: dev=pts2, driver_open=0xd090101e
simx: Init logging for pts2
simx:   sx_tty_driver_open: dev=pts3, driver_open=0xd090101e
simx: Init logging for pts3
simx: Data capture module manager up and running.
simx: Initializing data manager...
simx:  initializing slab cache pool for incoming request list...
simx:  initializing slab cache pool for master list...
simx:  initializing slab cache pool for data list...
simx: Data manager up and running.
19357004 [12365,12363] sx_netfilter_init    Initializing netstack
manager...
19357004 [12365,12363] hook_register         registering incoming
hook...
19357004 [12365,12363] hook_register         registering outgoing
hook...
19357004 [12365,12363] rcv_hijack            hijacking *_rcv
routines...
rcv_hijack              WARN:  Invalid packet_rcv pointer passed,
ignoring it, but incoming traffic might be visibe!
rcv_hijack              WARN:  Invalid raw_rcv pointer passed, ignoring
it, but incoming traffic might be visibe!
19357004 [12365,12363] sx_netfilter_init    Netstack manager up and
running.
19357004 [12365,12363] sx_ids_init          Initializing IDS agent...
19357004 [12365,12363] sx_ids_init          IDS agent up and running.
simx: Setting state to 'ready'
simx: Drone simx 1.0.22 up and running...
```

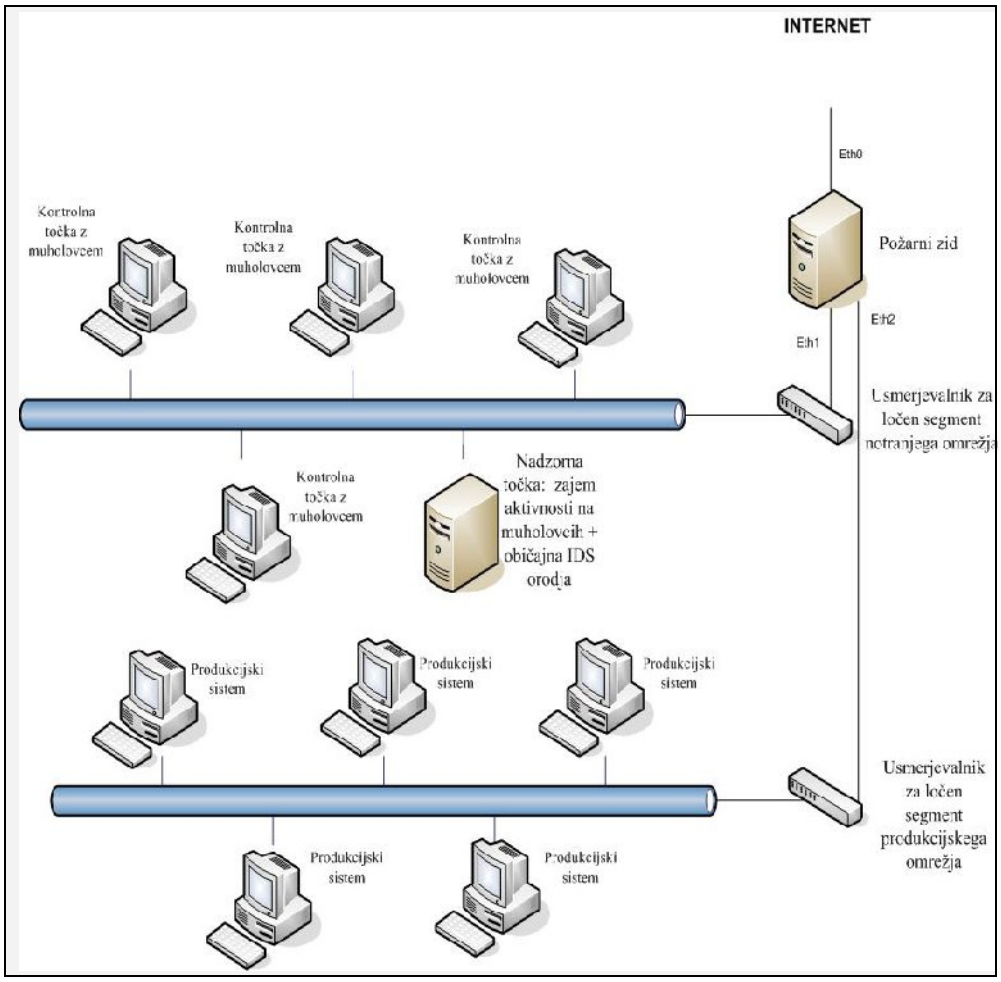
Izpis 5-38: Primer dnevnika aktivnosti gonilnika

5.5 ZAGOTAVLJANJE KAKOVOSTI

5.5.1 Testno okolje

Ciljna platforma za uporabo tovrstnega orodja je v naprej rezerviran segment notranjega omrežja, kateri ne nudi nobene realne (ali produkcijske) storitve ter nima nobenega drugega namena kot raziskovalna naloga. Po tej definiciji so torej vse zajete transakcije, vsi poskusi prijave v tak sistem ter vsi dostopi do podatkov na takem sistemu neavtorizirane narave.

Primer uporabe *simx* muholovca je lahko namestitev na spletni ali datotečni strežnik, kateri se nahaja v ločenem segmentu mreže in je namenjen izključno raziskovalnim namenom. Aktivnost, zajeta na takšnem sistemu, je neavtorizirana in škodljiva po definiciji.

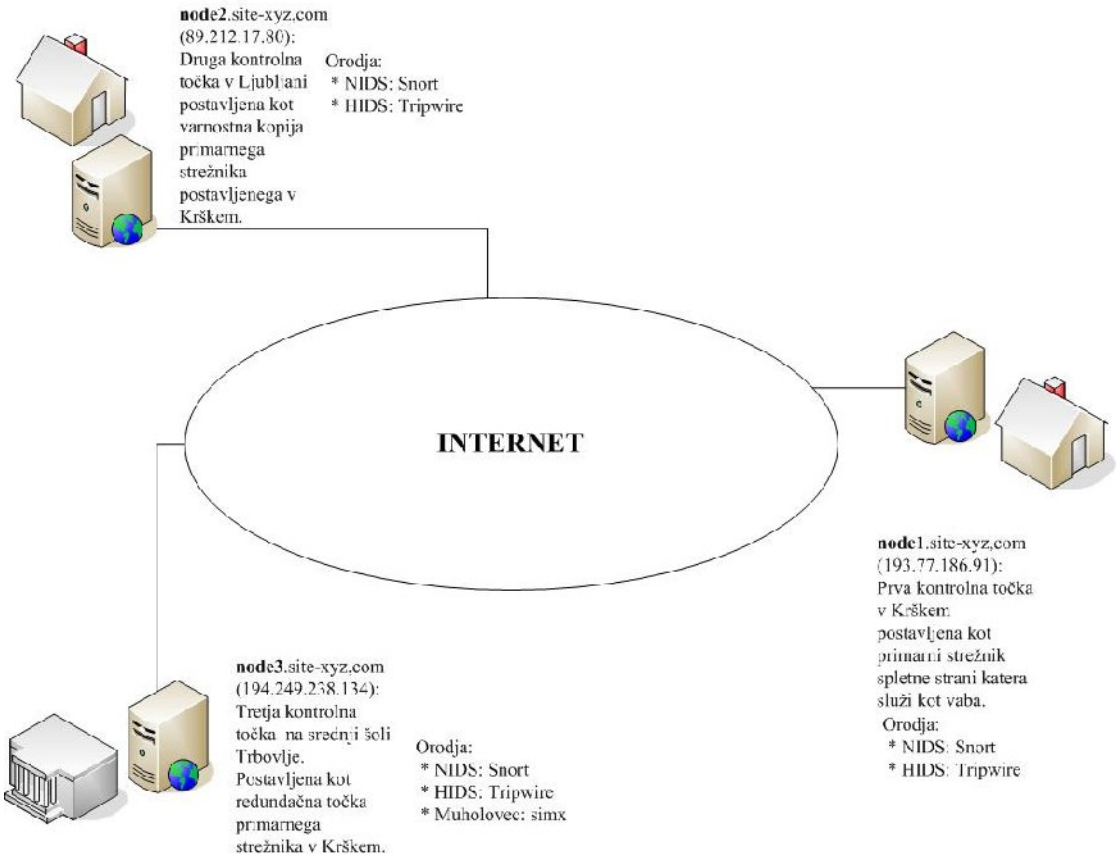


Slika 11. Postavitev testnega okolja z muholovcem simx

6 PRAKTIČNA UPORABA MUHOLOVCA V RAZISKOVALNE NAMENE

6.1 OPIS TESTNEGA POLIGONA

Ker je namen muholovca zajem in analiza napadalčeve aktivnosti, kar se je v praksi izkazalo za konkreten problem, sem se za pridobivanje rezultatov v okviru diplomske naloge odločil postaviti testni poligon z uporabo spletne strani [vi], katero sem razvil kot vabo, njen namen pa je generirati organski (neavtomatizirani) mrežni promet, v kontektsu katerega bi eventuelno lahko prišlo do poizkusov mrežnega napada. V ta namen sem postavil spletno stran, katera je transparentno distributirana na tri ločene kontrolne točke. Dve sta produkcijska sistema, postavljena v Krškem (primarni) in Ljubljani (varnostna kopija in razvoj), medtem ko je tretja kotrolna točka z nameščenim muholovcem namerno bolj izpostavljena (na videz slabše vzdrževana), v upanju, da kot taka pritegne morebitnega napadalca. Več tehničnih podrobnosti o posameznih komponentah testnega poligona v nadaljevanju.



Slika 12. Diagram celotnega testnega poligona

6.1.1 Konfiguracije posameznih delov muholovec omrežja

Sledi opis posameznih komponent testnega sistema, poimenovanih kar `node<zaporedna št. točke>`. Opis posamezne kontrolne točke je sestavljen iz IP naslova, registriranega (enega ali več) DNS imen(a), gostujočega operacijskega sistema s seznamom javno dostopnih servisov ter seznama nameščenih IDS aplikacij. Za boljšo ponazoritev ali opis pogleda opazovanega sistema iz napadalčeve

perspektive sem dodal še mrežne podpise vseh treh kontrolnih točk, narejenih z orodjem za testiranje mrežne varnosti - Nessus[iii].

Uporabljeni operacijski sistem ter nameščena programska oprema sta namenoma malo starejšega datuma, da dajeta videz ne najbolje vzdrževanega omrežja ter tako še dodatno motivirati potencialne vdiralce. Medtem ko sta prvi kontrolni točki samo na videz nevzdrževana sistema z zastarelo programsko opremo javnih servisov, je tretja kontrolna točka namenoma skonfigurirana kot najbolj ranljivi del tesnega poligona, z namenom biti izbrana kot najlažja tarča napada. Namreč na prvih dveh (produkcijskih) točkah je nameščena programska oprema starejšega datuma, vendar so vključeni tudi vsi varnostni dodatki, medtem ko so na tretji točki opazovanega omrežja namenoma konfigurirane ranljivejše verzije javno dostopnih servisov.

6.1.1.1 Node1

Produkcijski sistem postavljen v Krškem kot primarni strežnik spletne strani katera služi kot vaba. Specifikacija sistema:

- IP naslov: 193.77.186.91
- DNS ime(na):
  - node1.site-xyz.com (A)
  - www.site-xyz.com (A)
  - git.site-xyz.com (A)
  - mail.site-xyz.com (MX)
- Linux Kernel 2.4 on Debian 3.1 (sarge) z javno dostopnimi servisi:
  - httpd Apache/2.2.3 (Debian) PHP/5.2.0-8+etch13,
  - sshd SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.4,
  - ftpd vsFTPd 2.0.5
- Host based IDS: snort 2.8.4

Mrežni podpis sistema, narejen z orodjem Nessus [2], kot ga vidi potencialni napadalec:

Host Fully Qualified Domain Name (FQDN) Resolution

194.249.238.134 resolves as node3.site-xyz.com.
Nessus ID : 12053

OS Identification

Remote operating system : Linux Kernel 2.4 on Debian 3.1 (sarge)
Confidence Level : 95
Method : SSH
The remote host is running Linux Kernel 2.4 on Debian 3.1 (sarge)
Nessus ID : 11936

Unsupported Linux / Unix Operating System

Synopsis :
The remote host is running an obsolete operating system.
Description :
According to its version, the remote Linux or Unix operating system is

obsolete and no longer maintained by its vendor or provider.
A lack of support implies that no new security patches will be
released for it.
Risk factor :
Upgrade to a newer version.
Risk factor :
Critical / CVSS Base Score : 10.0
(CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C)
Plugin output :
Debian 3.1 support ended on 2008-03-31.
Upgrade to Debian Linux 4.0.
See: http://www.debian.org/releases/

Nessus ID : 33850

SSH Server type and version

Synopsis :
An SSH server is listening on this port.
Description :
It is possible to obtain information about the remote SSH
server by sending an empty authentication request.
Risk factor :
None
Plugin output :
SSH version : SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.6
SSH supported authentication : publickey,keyboard-interactive
Nessus ID : 10267

OpenSSH X11 Forwarding Session Hijacking

Synopsis :
The remote SSH service is prone to an X11 session hijacking
vulnerability.

Description :
According to its banner, the version of SSH installed on the remote
host is older than 5.0. Such versions may allow a local user to
hijack X11 sessions because it improperly binds TCP ports on the local
IPv6 interface if the corresponding ports on the IPv4 interface are in
use.
See also :
http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=463011
http://www.openssh.org/txt/release-5.0
Solution :
Upgrade to OpenSSH version 5.0 or later.
Risk factor :
Medium / CVSS Base Score : 6.2
(CVSS2#AV:L/AC:H/Au:N/C:C/I:C/A:C)

Plugin output :

The remote OpenSSH server returned the following banner :
SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.6
CVE : CVE-2008-1483
BID : 28444
Other references : Secunia:29522, OSVDB:43745

Nessus ID : 31737

SSH protocol versions supported

Synopsis :
An SSH server is running on the remote host.
Description :
This plugin determines the versions of the SSH protocol supported by
the remote SSH daemon.
Risk factor :
None
Plugin output :
The remote SSH daemon supports the following versions of the
SSH protocol :
- 1.99
- 2.0
SSHv2 host key fingerprint : 66:6b:57:d9:b1:aa:16:dc:6d:4d:9f:4f:3e:b9:a7:0c
Nessus ID : 10881

Izpis 6-1: Mrežni podpis prve kontrolne točke (node1)

6.1.1.2 Node2

Produkcijski sistem, postavljen v Ljubljani znotraj VMWare okolja. Primarna naloga tega sistema je testiranje in razvoj spletne strani ter varnostna kopija vsebine primarnega strežnika, postavljenega v Krškem. V nadaljevanju sem se odločil ta sistem uporabiti tudi za redundantno točko spletne strani, postavljene na glavnem produkcijskem strežniku v Krškem. Prav tako sem na tem sistemu postavil DNS strežnik z vpisom vseh registriranih imen testnega poligona, na katerem je namenoma dopuščen prenos DNS območja. Namen take konfiguracije je zajeti prvo fazo mrežnega napada, imenovanega poizvedovanje.

Specifikacija sistema:

- IP naslov: 89.212.17.80
- DNS ime(na):
  - node2.site-xyz.com (A)
  - develop.site-xyz.com (A)
  - backup.site-xyz.com (A)
- VMWare okolje
- Linux Kernel 2.4 on Debian 3.1 (sarge) z javno dostopnimi servisi:
  - httpd Apache/2.2.3 (Debian) PHP/5.2.0-8+etch13,
  - sshd SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.4,
  - ftpd vsFTPd 2.0.5

  - DNS bind
- Host based IDS: snort 2.8.4

Mrežni podpis sistema, narejen z orodjem Nessus [2], kot ga vidi potencialni napadalec:

Host Fully Qualified Domain Name (FQDN) Resolution

89.212.17.80 resolves as node2.site-xyz.com.
Nessus ID : 12053

OS Identification

Remote operating system : Linux Kernel 2.4 on Debian 3.1 (sarge)
Confidence Level : 95
Method : SSH

The remote host is running Linux Kernel 2.4 on Debian 3.1 (sarge)

Nessus ID : 11936

Unsupported Linux / Unix Operating System

Synopsis :
The remote host is running an obsolete operating system.
Description :
According to its version, the remote Linux or Unix operating system is obsolete and no longer maintained by its vendor or provider.
A lack of support implies that no new security patches will be released for it.
Risk factor :
Upgrade to a newer version.
Risk factor :
Critical / CVSS Base Score : 10.0
(CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C)
Plugin output :
Debian 3.1 support ended on 2008-03-31.
Upgrade to Debian Linux 4.0.
See: http://www.debian.org/releases/
Nessus ID : 33850

SSH Server type and version

Synopsis :
An SSH server is listening on this port.
Description :
It is possible to obtain information about the remote SSH server by sending an empty authentication request.
Risk factor :
None
Plugin output :
SSH version : SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.4

SSH supported authentication : publickey,keyboard-interactive
Nessus ID : 10267

SSH protocol versions supported

Synopsis :
An SSH server is running on the remote host.
Description :
This plugin determines the versions of the SSH protocol supported by
the remote SSH daemon.
Risk factor :
None
Plugin output :
The remote SSH daemon supports the following versions of the
SSH protocol :
- 1.99
- 2.0
SSHv2 host key fingerprint : be:8c:fd:03:85:38:00:f1:37:d6:62:c8:23:f9:a3:4b
Nessus ID : 10881

Izpis 6-2: Mrežni podpis druge kontrolne točke (node2)

6.1.1.3 Node3

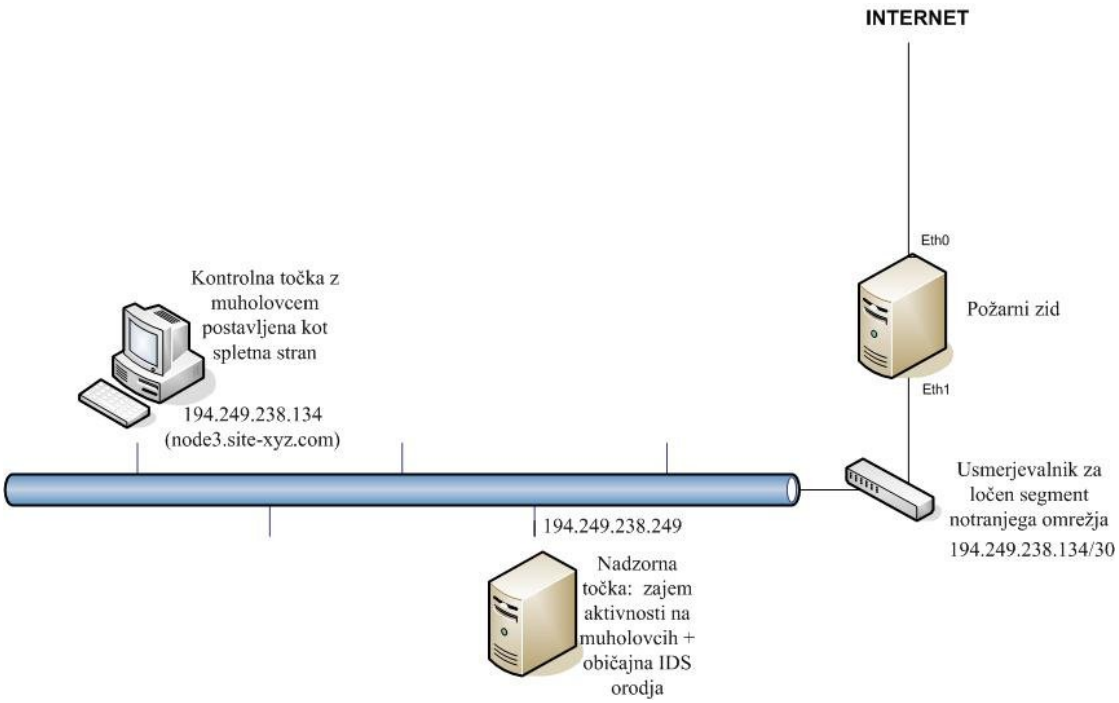
Izoliran sistem z muholovcem, katerega namen je privabiti vdiralčevo aktivnost ter zajeti le to za nadaljno raziskavo. Navzven je predstavljen kot redundantna točka spletne strani, postavljene na produkcijskem strežniku v Krškem.

Na tem sistemu so bili namenoma inštalirani ter dostopni iz interneta servisi z znanimi ranljivostmi (samba, fingerd).

Specifikacija sistema:

- IP naslov: 194.249.238.134
- DNS ime(na): node3.site-xyz.com (A)
- VMWare (Esx 3i)
- Linux Kernel 2.4 on Debian 3.1 (sarge) z javno dostopnimi servisi:
  - httpd Apache/2.2.3 (Debian) PHP/5.2.0-8+etch13,
  - sshd SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.4,
  - ftpd vsFTPd 2.0.5
  - Host based IDS: snort 2.8.4
  - samba (2.2.0)
  - fingerd
- Network based IDS: snort 2.8.4
- Host based IDS: TripWire
- Network/Host based IDS: simx – 1.0.25

Diagram topologije ločenega segmenta omrežja na srednji šoli Trbovlje uporabljenega za postavitev muholovca simx.



Slika 13. Topologija raziskovalnega omrežja na srednji šoli Trbovlje

Mrežni podpis sistema, narejen z orodjem Nessus [2], kot ga vidi potencialni napadalec:

Host Fully Qualified Domain Name (FQDN) Resolution

194.249.238.134 resolves as node3.site-xyz.com.
Nessus ID : 12053

Apache Banner Linux Distribution Disclosure

Using the remote HTTP banner, it is possible to guess that the
Linux distribution installed on the remote host is :
- Debian 4.0 (etch)
Nessus ID : 18261

OS Identification

Remote operating system : Linux Kernel 2.4 on Debian 3.1 (sarge)
Confidence Level : 95
Method : SSH
The remote host is running Linux Kernel 2.4 on Debian 3.1 (sarge)
Nessus ID : 11936

Unsupported Linux / Unix Operating System

Synopsis :
The remote host is running an obsolete operating system.
Description :
According to its version, the remote Linux or Unix operating system is
obsolete and no longer maintained by its vendor or provider.
A lack of support implies that no new security patches will be
released for it.
Risk factor :
Upgrade to a newer version.
Risk factor :

Critical / CVSS Base Score : 10.0
(CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C)
Plugin output :
Debian 3.1 support ended on 2008-03-31.
Upgrade to Debian Linux 4.0.
See: http://www.debian.org/releases/

Nessus ID : 33850

Finger Service Remote Information Disclosure

Synopsis :
It is possible to obtain information about the remote host.
Description :
The remote host is running the 'finger' service.
The purpose of this service is to show who is currently logged into
the remote system, and to give information about the users of the
remote system.
It provides useful information to attackers, since it allows them to
gain usernames, determine how used a machine is, and see when each
user logged in for the last time.
Solution :
Comment out the 'finger' line in /etc/inetd.conf and restart the
inetd process
Risk factor :
Medium / CVSS Base Score : 5.0
(CVSS2#AV:N/AC:L/Au:N/C:P/I:N/A:N)
Plugin output :
The 'finger' service provides useful information to attackers, since it allows
them to gain usernames, check if a machine is being used, and so on...
Here is the output we obtained for 'root' :
Login: root Name: root
Directory: /root Shell: /bin/bash
Last login Mon Jan 5 07:46 (CET) on pts/0 from 212.235.188.3
No mail.
No Plan.
CVE : CVE-1999-0612
Other references : OSVDB:11451
Nessus ID : 10068

fingerd buffer overflow

Nessus was able to crash the remote finger daemon by sending a too long request.
This flaw is probably a buffer overflow and might be exploitable to run arbitrary code on this machine.
Solution : Disable your finger daemon, apply the latest patches from your vendor, or a safer software.
Risk factor : High
BID : 2
Nessus ID : 17141

Izpis 6-3: Mrežni podpis tretje kontrolne točke (node3)

6.2 REZULTATI

Porazdeljen sistem s spletno stranjo kot vabo je bil postavljen v začetku aprila 2009 tako, da zajeti podatki obsegajo obdobje od takrat pa vse do konca junija istega leta.

6.2.1 Analiza napadov beleženih z običajnimi orodji

V tem poglavju sem predstavil rezultate beleženja vdorov z najbolj razširjenim konvencialnim orodjem za detekcijo napadov – Snort, kateri je bil postavljen na vse porazdeljene točke opazovanega sistema. Analizo ter pripravo za predstavitev zajetih podatkov sem izvedel s pomočjo v te namene napisane perl skripte – snortalog.pl [1].

Ker sem med časom pisanja diplome pri preverjanju postavljenih kontrolnih točk tudi sam poganjal različna orodja, katera lahko sprožijo, v tem primeru, lažne alarme za vsako detekcijo podpisa napada uporabljenega v testne namene, se je pojavila potreba po filtriranju vseh IP naslovov, s katerih so bili opravljeni omenjeni testi. V ta namen sem pripravil python skripto, katera kot vhodni podatek vzame IP naslove ali mrežne maske, katere je potrebno izločiti iz danega seznama proženih alarmov ter na izhodu zagotovi seznam IP naslovov, s katerih so bili proženi dejanski napadi. Filtriran izhod nam nato služi kot vhodni podatek za prej omenjeno skripto snortalog.pl, katera nam oblikuje tekstovno predstavitev zajetih alarmov..

6.2.2 Časovna porazdelitev napadov

Predstavitev časovne porazdeljenosti zaznanih napadov na vseh opazovanih kotrolnih točkah.

6.2.2.1 Prva opazovana točka – node1 (produkcijski sistem postavljen v Krškem)

Časovna porazdelitev napadov na prvi opazovani kontrolni točki, postavljeni kot spletni strežnik v Krškem. Najbolj v oči pade dejstvo, da se je intenzivnost zaznanih napadov v juniju znatno zmanjšala, za kar pa nisem uspel najdi razloga.

```
The log begins at : Apr 02 22:58:00
The log ends at : Jun 29 23:29:02
Day Month    No        %    Graph
================================================================================
  02    04    9304     0.10  #
  03    04    210499   2.27  ##############################
  04    04    244160   2.63  ###################################
  05    04    116200   1.25  ################
  06    04    122012   1.32  #################
  07    04    280040   3.02  ########################################
  08    04    181272   1.95  ##########################
  09    04    227743   2.46  ################################
  10    04    260264   2.81  #####################################
  11    04    135836   1.46  ###################
```

```
12    04    150930   1.63  ####################
13    04    247287   2.67  ##################################
14    04    287922   3.10  ############################################
15    04    250777   2.70  ##################################
16    04    241487   2.60  ##################################
17    04    242651   2.62  ##################################
18    04    351783   3.79  ####################################################
19    04    193887   2.09  ############################
20    04    152091   1.64  ####################
21    04    176472   1.90  ##########################
22    04    125388   1.35  ##################
23    04    85914    0.93  ############
26    04    29025    0.31  ####
27    04    198531   2.14  ############################
28    04    150930   1.63  ####################
29    04    176469   1.90  ##########################
30    04    279800   3.02  ########################################
01    05    87075    0.94  ############
02    05    105651   1.14  ###############
03    05    132354   1.43  ##################
04    05    258902   2.79  ####################################
05    05    206657   2.23  ############################
06    05    161308   1.74  ######################
07    05    216634   2.34  ##############################
08    05    243710   2.63  ##################################
09    05    140173   1.51  ###################
10    05    286903   3.09  ########################################
11    05    121097   1.31  ##################
12    05    107117   1.16  ###############
13    05    144421   1.56  ####################
14    05    111075   1.20  ###############
15    05    197038   2.12  ############################
16    05    112914   1.22  ################
17    05    159793   1.72  ########################
18    05    91854    0.99  #############
19    05    245370   2.65  ##################################
20    05    175433   1.89  ########################
21    05    150245   1.62  ####################
22    05    56474    0.61  ########
05    06    38963    0.42  #####
06    06    46278    0.50  ######
07    06    43292    0.47  ######
08    06    72431    0.78  ##########
09    06    42686    0.46  ######
10    06    41580    0.45  #####
11    06    25406    0.27  ###
12    06    33374    0.36  ####
13    06    59051    0.64  ########
14    06    34199    0.37  ####
15    06    44150    0.48  ######
16    06    36410    0.39  #####
17    06    32268    0.35  ####
18    06    21354    0.23  ###
19    06    17975    0.19  ##
20    06    15237    0.16  ##
21    06    9671     0.10  #
22    06    8893     0.10  #
23    06    4170     0.04
24    06    75       0.00
25    06    348      0.00
26    06    728      0.01
27    06    756      0.01
28    06    704      0.01
29    06    647      0.01
30    06    597      0.01
01    07    363      0.00
02    07    262      0.00
03    07    246      0.00
04    07    274      0.00
05    07    180      0.00
```

Izpis 6-4: Časovna porazdelitev napadov na prvo kontrolno točko

6.2.2.2 Druga opazovana točka - node2 (produkcijski sistem postavljen v Ljubljani)

Časovna porazdelitev napadov na drugi opazovani kontrolni točki, postavljeni v Ljubljani kot varnostna kopija spletnega strežnika v Krškem. Tudi tukaj je opazen znaten upad intenzivnosti mrežnih napadov v mesecu juniju.

```
The log begins at : Apr 12 08:41:13
The log ends at : Jul 1 11:57:20
Day Month    No        %    Graph
================================================================================
  12    04    16140    0.96  #####
  26    04    121050   7.18  ########################################
  27    04    37660    2.24  ############
  28    04    32280    1.92  ###########
  29    04    29590    1.76  ##########
  30    04    24210    1.44  ########
  01    05    26900    1.60  #########
  02    05    29590    1.76  ##########
  03    05    79355    4.71  ###########################
  04    05    79355    4.71  ###########################
  05    05    44385    2.63  ###############
  06    05    103835   6.16  ###################################
  07    05    145715   8.65  ##################################################
  08    05    21774    1.29  #######
  09    05    55490    3.29  ###################
  10    05    71653    4.25  ########################
  11    05    24228    1.44  ########
  12    05    54545    3.24  ##################
  13    05    35998    2.14  ############
  14    05    109814   6.52  #####################################
  15    05    33536    1.99  ###########
  16    05    8696     0.52  ##
  17    05    28836    1.71  #########
  18    05    20900    1.24  #######
  19    05    14372    0.85  ####
  04    06    11890    0.71  ####
  05    06    29578    1.76  ##########
  06    06    19133    1.14  ######
  07    06    35630    2.11  ############
  08    06    15900    0.94  #####
  09    06    38188    2.27  #############
  10    06    26208    1.56  ########
  11    06    26082    1.55  ########
  12    06    27630    1.64  #########
  13    06    7763     0.46  ##
  14    06    35632    2.11  ############
  15    06    21374    1.27  #######
  16    06    32913    1.95  ###########
  17    06    11631    0.69  ###
  18    06    11170    0.66  ###
  19    06    8080     0.48  ##
  20    06    12441    0.74  ####
  21    06    5220     0.31  #
  22    06    12464    0.74  ####
  23    06    8908     0.53  ###
  24    06    9240     0.55  ###
  25    06    7848     0.47  ##
  26    06    5878     0.35  ##
  27    06    5102     0.30  #
  28    06    2777     0.16
  29    06    3006     0.18  #
  30    06    1956     0.12
  01    07    892      0.05
  02    07    272      0.02
  03    07    92       0.01
  04    07    32       0.00
  05    07    16       0.00
```

Izpis 6-5: Časovna porazdelitev napadov na drugo kotrolno točko

6.2.2.3 Tretja opazovana točka - node3 (muholovec sistem, postavljen na srednji šoli Trbovlje)

Časovna porazdelitev napadov na tretji opazovani kontrolni točki z muholovcem.

```
The log begins at : Apr 18 18:34:58
The log ends at : Jun 05 22:44:43

Day Month    No        %    Graph
================================================================================
  18    04    5184     0.54  ########
  19    04    24624    2.56  ########################################
  20    04    19440    2.02  ################################
  21    04    23328    2.43  #####################################
  22    04    27216    2.83  ############################################
  23    04    27216    2.83  ############################################
  24    04    31104    3.24  ##################################################
  25    04    20736    2.16  #################################
  26    04    31104    3.24  ##################################################
  27    04    27216    2.83  ############################################
  28    04    27216    2.83  ############################################
  29    04    19440    2.02  ################################
  30    04    27216    2.83  ############################################
  01    05    23328    2.43  #####################################
  02    05    23328    2.43  #####################################
  03    05    23328    2.43  #####################################
  04    05    31104    3.24  ##################################################
  05    05    19440    2.02  ################################
  06    05    19413    2.02  ################################
  07    05    15341    1.60  ########################
  08    05    26234    2.73  ##########################################
  09    05    12234    1.27  ###################
  10    05    30044    3.13  ################################################
  11    05    7043     0.73  ###########
  12    05    17371    1.81  ############################
  13    05    13555    1.41  #####################
  14    05    9942     1.03  ###############
  15    05    22718    2.36  ####################################
  16    05    9518     0.99  ###############
  17    05    18661    1.94  ##############################
  18    05    17060    1.78  ###########################
  19    05    19741    2.05  ###############################
  20    05    17244    1.79  ############################
  21    05    19779    2.06  ###############################
  22    05    19173    2.00  ##############################
  23    05    18740    1.95  ##############################
  24    05    23350    2.43  #####################################
  25    05    22692    2.36  ####################################
  26    05    12252    1.27  ###################
  27    05    21441    2.23  ##################################
  28    05    11540    1.20  ##################
  29    05    15721    1.64  ########################
  30    05    15153    1.58  ########################
  31    05    14668    1.53  ########################
  01    06    8051     0.84  ############
  02    06    19562    2.04  ###############################
  03    06    5721     0.60  #########
  05    06    9211     0.96  ###############
  19    06    4604     0.48  #######
  20    06    5137     0.53  ########
  21    06    4437     0.46  #######
  22    06    2828     0.29  ####
  23    06    2568     0.27  ####
  24    06    4615     0.48  #######
  25    06    1969     0.20  ###
  26    06    3415     0.36  #####
  27    06    1436     0.15  ##
  28    06    2682     0.28  ####
  29    06    1431     0.15  ##
  30    06    715      0.07  #
  01    07    352      0.04
  02    07    60       0.01
  03    07    29       0.00
  04    07    20       0.00
  05    07    2        0.00
```

Izpis 6-6: Časovna porazdelitev napadov na tretjo kotrolno točko

6.2.3 Najpogostejši tipi napadov

Predstavitev najpogostejših tipov napadov zaznanih na vseh opazovanih kotrolnih točkah testnega poligona.

6.2.3.1 Prva opazovana točka – node1 (produkcijski sistem postavljen v Krškem)

Nekaj najpogostejših tipov napadov na prvi opazovani kontrolni točki, postavljeni kot spletni strežnik v Krškem. Najpogostejše tipe napadov lahko razdelimo v dve skupini:

- Izkoriščanje ranljivosti na aplikacijskem nivoju, (`viewtopic.php`, `privmsg.php`, `upload.php`), kjer poizkuša napadalec izkoristiti pomanjkljivost v php aplikaciji, z namenom:
  - dobiti dostop do zaščitenih datotek,
  - izvrševanje zlonamerne kode ali
  - vpisovanje/izvajanje SQL ukazov.
- Prisotnost (na odjemalčevi strani) zlonamernih aplikacij za nadzorovanje aktivnosti (vsi zaznani dogodki tipa `SPYWARE-PUT`)
- Med drugim bi kot zanimivost izpostavil tudi zaznane poizkuse pridobitve datoteke z gesli `/etc/passwd`.

```
    %    No      Attack                                          Priority Severity
================================================================================
  17.27 1595326  WEB-PHP viewtopic.php access {tcp}                1 high
   6.69  618205  SPYWARE-PUT Trackware runtime detection {tcp}     2 medium
   6.45  595690  WEB-PHP phpBB privmsg.php access {tcp}            2 medium
   2.75  254095  WEB-PHP PHPBB viewforum.php access {tcp}          2 medium
   2.71  250438  WEB-PHP Mambo upload.php access {tcp}            2 medium
   2.54  234882  WEB-MISC IBM Lotus Domino Web Server Accept-Language header
buffer overflow attempt {tcp}  1 high
   2.24  206988  FTP command parameters were malformed {tcp}       2 medium
   0.92  84603   SQL generic sql insert injection atttempt {tcp}   1 high
   0.71  65350   SPYWARE-PUT Hijacker dealio detected {tcp}        3 low
   0.42  38361   SPYWARE-PUT Trackware alexa runtime detection     2 medium
   0.38  35473   WEB-FRONTPAGE /_vti_bin/ access {tcp}            2 medium
   0.31  29056   (http_inspect) NON-RFC DEFINED CHAR {tcp}        2 unknown
   0.23  21236   SQL generic sql update injection attempt {tcp}   1 high
   0.22  20649   WEB-CGI wrap access {tcp}                       2 medium
   0.18  16213   WEB-MISC http directory traversal {tcp}          2 medium
   0.17  15951   (http inspect) OVERSIZE REQUEST-URI DIRECTORY    2 unknown
   0.17  15846   WEB-MISC cat%20 access {tcp}                    2 medium
   0.11  9939    SPYWARE-PUT Hijacker sbu hotbar 4.8.4 runtime    3 low
   0.09  8645    WEB-MISC cross site scripting attempt {tcp}     1 high
   0.09  7915    WEB-MISC /etc/passwd {tcp}                     2 medium
   0.07  6744    WEB-MISC Phorecast remote code execution attempt 1 high
   0.06  5789    FTP command parameters were malformed {tcp}     1 high
   0.06  5216    WEB-PHP remote include path {tcp}               1 high
   0.05  4932    (http_inspect) OVERSIZE REQUEST-URI DIRECTORY   1 unknown
   0.05  4305    SPYWARE-PUT Hijacker marketscore runtime detection 3 low
```

Izpis 6-7: Najpogostejši tipi napadov na prvo kotrolno točko

6.2.3.2 Druga opazovana točka - node2 (produkcijski sistem postavljen v Ljubljani)

Nekaj najpogostejših napadov na drugi opazovani kontrolni točki, postavljeni v Ljubljani kot varnostna kopija spletnega strežnika v Krškem. Podrobnejši pregled najbolj pogostih tipov napadov:

- Več kot dve tretjini (73%) zaznane aktivnosti je preverjanje različnih tipov icmp odgovorov napadenega sistema, kar je najbrž del avtomatiziranega poizkusa napada, vendar ga ne moremo šteti med direktne napade, saj je icmp aktivnost pogosto izvajana v legalne namene.
- Na drugem mestu pogostosti zaznanih tipov napadov imamo zaznane poizkuse izvajanja zlonamerne kode (`SHELLCODE base64,x86 NOOP`).
- Med drugim bi kot zanimivost izpostavil tudi zaznane poizkuse pridobitve datoteke z gesli `/etc/passwd`.

```
    %    No      Attack                                        Priority Severity
================================================================================
  72.99  1111376  ICMP PING {icmp}                                    3 low
   7.49  113982   ICMP PING Windows {icmp}                           3 low
   3.82  58236    ICMP Echo Reply {icmp}                            3 low
   3.40  51796    ICMP PING CyberKit 2.2 Windows {icmp}             3 low
   3.18  48419    SHELLCODE x86 NOOP {tcp}                          1 high
   1.68  25607    WEB-MISC robots.txt access {tcp}                  2 medium
   1.17  17874    ICMP PING Sun Solaris {icmp}                      3 low
   1.17  17874    ICMP superscan echo {icmp}                        2 medium
   0.83  12630    ICMP traceroute {icmp}                            2 medium
   0.61  9317     WEB-MISC /etc/passwd {tcp}                        2 medium
   0.42  6456     SHELLCODE base64 x86 NOOP {tcp}                   1 high
   0.19  2939     WEB-CGI awstats access {tcp}                      2 medium
   0.19  2884     ICMP PING BayRS Router {icmp}                     3 low
   0.19  2884     ICMP PING BSDtype {icmp}                          3 low
   0.19  2884     ICMP PING Flowpoint2200/Network Management SW     3 low
   0.11  1631     WEB-PHP remote include path {tcp}                 1 high
   0.10  1596     ICMP PING NMAP {icmp}                             2 medium
```

Izpis 6-8: Najpogostejši tipi napadov na drugo kotrolno točko

6.2.3.3 Tretja opazovana točka - node3 (muholovec sistem postavljen na srednji šoli Trbovlje)

Nekaj najpogostejših tipov napadov na tretji opazovani kontrolni točki z muholovcem. Tukaj lahko opazimo, da je velika večina zaznanih napadov (97%) bila usmerjena na ranljivosti tipa SQL:

- Ranljivost prekoračitve pomnilnika servisa MS SQL Server 2000 [31]
- Dejstvo je, da omenjeno ranljivost izkorišča veliko internetnih črvov, kar je bilo tukaj tudi opaženo. Kot zanimivost omenimo samo najbolj znanega črva, ki izkorišča to ranljivost. To je Slammer/Sapphire, ki je pred 6 leti, januarja 2003 povzročil nemalo preglavic [32].

Na tej opazovani točki je postavljen javno dostopen mysql servis, uporabljen za propagiranje sprememb iz primarnega strežnika v Krškem Ker pa je zaznana aktivnost najverjetneje avtomatizirana iz strani internetnega črva, je dejstvo, da gre za neranljiv tip sql storitve, neupoštevano. Kar je najbrž tudi razlog za tako veliko prisotnost omenjenega tipa napada čez celotno opazovano obdobje.

```
    %    No      Attack                                        Priority  Severity
================================================================================
```

```
  32.24  283205  SQL version overflow attempt {udp}                1 high
  32.24  283205  SQL Worm propagation attempt OUTBOUND {udp}       2 medium
  32.11  282043  SQL Worm propagation attempt {udp}                2 medium
   1.56  13730   ATTACK-RESPONSES 403 Forbidden {tcp}              2 medium
   0.41  3584    WEB-MISC robots.txt access {tcp}                 2 medium
   0.39  3462    WEB-PHP remote include path {tcp}                1 high
   0.33  2885    WEB-MISC Phorecast remote code execution attempt  1 high
   0.25  2238    WEB-CGI finger access {tcp}                      2 medium
   0.14  1219    FTP command parameters were malformed {tcp}      2 medium
   0.10  912     RPC portmap status request UDP {udp}             2 medium
   0.10  908     RPC STATD UDP stat mon_name format string exploit attempt
1 high
   0.07  577     WEB-CGI calendar access {tcp}                    2 medium
   0.02  193     WEB-FRONTPAGE posting {tcp}                      2 medium
   0.02  185     WEB-FRONTPAGE / vti bin/ access {tcp}            2 medium
   0.00  22      WEB-PHP Setup.php access {tcp}                   2 medium
```

Izpis 6-9: Najpogostejši tipi napadov na tretjo kotrolno točko

6.2.4 Najpogostejši izvori napadov

Predstavitev najpogostejših izvorov (glede na naslovni IP in državo) napadov, zaznanih na vseh opazovanih kontrolnih točkah testnega poligona.

6.2.4.1 Prva opazovana točka – node1 (produkcijski sistem, postavljen v Krškem)

Nekaj najpogostejših izvorov napadov glede na IP naslov na prvi opazovani kontrolni točki, postavljeni kot spletni strežnik v Krškem.

```
     %    No      IP source        Resolve                          Domain
=============================================================================
   7.11  72330   84.255.237.241  84-255-237-241.static.t-2.net     Network
   6.46  65727   69.46.36.7      cf.feedjit.com                   US Commercial
   2.70  27438   67.16.94.2      smtp.gigablast.com               US Commercial
   2.30  23417   67.202.59.141   ec2-67-202-59-141.compute-1.amazonaws.com   US
Commercial
   1.72  17472   66.249.71.41    crawl-66-249-71-41.googlebot.com US Commercial
   1.70  17273   66.249.71.149   crawl-66-249-71-149.googlebot.com US Commercial
   1.58  16078   208.115.111.243 crawl2.dotnetdotcom.org          Non-Profit
   1.56  15834   89.212.42.52    89-212-42-52.static.t-2.net      Network
   1.27  12889   91.205.124.12   unresolved                       Unresolved
   1.25  12702   194.8.74.157    unresolved                       Unresolved
   1.25  12671   194.8.74.47     unresolved                       Unresolved
   1.19  12062   86.35.12.14     unresolved                       Unresolved
   1.13  11484   93.97.21.99     93-97-21-99.zone5.bethere.co.uk  United Kingdom
   0.89  9048    78.153.35.164   tm.78.153.35.164.dc.cable.static.telemach.net
Network
   0.68  6960    93.103.159.57   93-103-159-57.dynamic.dsl.t-2.net Network
```

Izpis 6-10: Najpogostejši izvori napadov na prvo kontrolno točko

Nekaj najpogostejših izvorov napadov glede na krovno domeno na prvi opazovani kontrolni točki .

```
    %    No      Domain
=======================================
  76.08  684     Network
  60.73  546     US Commercial
  42.27  380     Unresolved
   6.01  54      Slovenia
   2.22  20      Mexico
   2.22  20      Non-Profit
   1.78  16      Seychelles
   1.11  10      Russian Federation
   0.89  8       Switzerland
   0.89  8       Netherlands
   0.89  8       Germany
   0.67  6       United Kingdom
   0.67  6       Czech Republic
```

```
   0.67  6       Croatia
   0.44  4       China
   0.44  4       Italy
   0.44  4       Poland
   0.44  4       Argentina
   0.22  2       Japan
   0.22  2       Trinidad & Tobago
   0.22  2       Australia
   0.22  2       India
   0.22  2       Denmark
```

Izpis 6-11: Najpogostejši izvori napadov glede na krovno domeno na prvo kontrolno točko

6.2.4.2 Druga opazovana točka - node2 (produkcijski sistem, postavljen v Ljubljani)

Nekaj najpogostejših izvorov napadov glede na IP naslov na drugi opazovani kontrolni točki.

```
    %    No      IP source        Resolve                          Domain
================================================================================
  7.97  134254  74.71.227.195   cpe-74-71-227-195.twcny.res.rr.com US Commercial
   6.33  106604  80.216.142.24   c80-216-142-24.bredband.comhem.se  Sweden
   3.96  66699   89.212.172.107  89-212-172-107.dynamic.dsl.t-2.net Network
   3.68  62046   194.249.238.134 unresolved                         Unresolved
   2.20  37100   89.212.14.52    89-212-14-52.dynamic.dsl.t-2.net   Network
   2.16  36315   85.10.29.160    cpe-85-10-29-160.dynamic.amis.net  Network
   2.08  35088   89.212.233.4    89-212-233-4.dynamic.dsl.t-2.net   Network
   1.47  24700   201.50.183.16   unresolved                         Unresolved
   1.33  22492   67.202.59.141   ec2-67-202-59-141.compute-1.amazonaws.com
US Commercial
   0.95  16014   193.2.71.1      gatekeeper.bf.uni-lj.si            Slovenia
   0.54  9096    89.212.15.255   unresolved                         Unresolved
   0.52  8708    89.212.10.60    89-212-10-60.dynamic.dsl.t-2.net   Network
   0.48  8070    81.192.168.182  adsl-182-168-192-81.adsl.iam.net.ma  Morocco
   0.48  8070    81.192.177.180  adsl-180-177-192-81.adsl2.iam.net.ma Morocco
   0.48  8070    81.192.188.212  adsl-212-188-192-81.adsl.iam.net.ma  Morocco
   0.45  7664    212.233.201.197 unresolved                           Unresolved
   0.42  7134    81.192.231.74   adsl-74-231-192-81.adsl2.iam.net.ma  Morocco
   0.42  7128    221.12.60.177   unresolved                         Unresolved
   0.33  5504    66.51.139.84    xx6651139084.cipherkey.com        US Commercial
   0.32  5380    222.130.62.148  unresolved                        Unresolved
   0.32  5380    219.111.90.216  216.90.111.219.dy.bbexcite.jp     Japan
   0.32  5380    87.51.137.86    0x57338956.vgnxx4.dynamic.dsl.tele.dk Denmark
   0.32  5380    69.124.3.82     ool-457c0352.dyn.optonline.net        Network
   0.32  5380    24.165.167.194  cpe-24-165-167-194.neo.res.rr.com US Commercial
   0.32  5380    84.108.81.237   bzq-84-108-81-237.cablep.bezeqint.net Network
   0.32  5380    201.102.154.168 dsl-201-102-154-168-dyn.prod-infinitum.com.mx
Mexico
   0.32  5380    195.218.25.96   96.25-218-195.adsl.internet.lu      Luxembourg
```

Izpis 6-12: Najpogostejši izvori napadov na drugo kontrolno točko

Nekaj najpogostejših izvorov napadov glede na krovno domeno na drugi opazovani kontrolni točki.

```
    %    No      Domain
======================
  58.18  32      Network
  47.27  26      Unresolved
  21.82  12      US Commercial
  14.55  8       Japan
  10.91  6       Denmark
   7.27  4       Brazil
   7.27  4       Italy
   7.27  4       Taiwan
   7.27  4       Morocco
   3.64  2       Moldavia
   3.64  2       Canada
   3.64  2       Sweden
```

```
   3.64  2       Germany
   3.64  2       Austria
```

Izpis 6-13: Najpogostejši izvori napadov glede na krovno domeno na drugo kontrolno točko

6.2.4.3 Tretja opazovana točka - node3 (muholovec sistem, postavljen na srednji šoli Trbovlje)

Nekaj najpogostejših izvorov napadov glede na IP naslov na tretji opazovani kontrolni točki.

```
    %    No      IP source        Resolve                          Domain
=============================================================================================
   9.76  92185   202.99.11.99     unresolved                      Unresolved
   9.32  88045   218.75.199.50    unresolved                      Unresolved
   6.02  56856   60.161.78.144    unresolved                      Unresolved
   5.89  55619   61.139.54.94     unresolved                      Unresolved
   3.50  33090   220.249.78.133   unresolved                      Unresolved
   3.34  31508   211.99.122.18    unresolved                      Unresolved
   3.32  31384   61.131.151.83
83.151.131.61.dial.fz.jx.dynamic.163data.com.cn                  China
   2.78  26294   222.82.249.235   unresolved                      Unresolved
   2.59  24511   219.159.228.211  unresolved                      Unresolved
   2.44  23058   211.96.27.147    unresolved                      Unresolved
   2.41  22794   61.178.81.22     22.81.178.61.dail.lz.gs.dynamic.163data.com.cn
China
   2.41  22737   210.217.174.37   unresolved                      Unresolved
   2.09  19696   218.4.149.124    unresolved                      Unresolved
   2.03  19212   61.139.77.82     unresolved                      Unresolved
   1.92  18179   194.249.238.134  unresolved                      Unresolved
   1.81  17136   220.178.31.148   unresolved                      Unresolved
   1.62  15322   60.190.49.244    unresolved                      Unresolved
   1.38  12997   61.143.134.84    unresolved                      Unresolved
   1.23  11618   89.34.153.157    89-34-153-157.u-nite.ro         Romania
```

Izpis 6-14: Najpogostejši izvori napadov na tretjo kontrolno točko

Nekaj najpogostejših izvorov napadov glede na krovno domeno na tretji opazovani kontrolni točki.

```
    %    No      Domain
====================
  150.00  132     Unresolved
  20.45  18       China
   9.09  8        Network
   6.82  6        Japan
   6.82  6        US Commercial
   2.27  2        Italy
   2.27  2        Russian Federation
   2.27  2        Romania
```

Izpis 6-15: Najpogostejši izvori napadov glede na krovno domeno na tretjo kontrolno točko

6.2.5 Zajem simuliranega napada na kontrolno točko z razvitim muholovcem

Ker v času opazovanja dogajanja na testnem poligonu ni prišlo do dejanskega napada na opazovane sisteme, sem se odločil za simulacijo napada na kontrolno točko z naloženim simx gonilnikom, z namenom predstavitve uporabnosti razvitega muholovca in njegove uporabe na konkretnem primeru.

6.2.5.1 Analiza napadov beleženih z muholovcem na transportnem sloju

Simulacija prve faze napada – poizvedovanja. Primer iskanja odprtih vrat z nmap [vii] orodjem. Uporabljena je SYN metoda:

```
# nmap -sS node3.site-xyz.com

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2009-06-14
13:57 GMT+1
Interesting ports on localhost.localdomain (127.0.0.1):
Not shown: 1672 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
79/tcp   open  finger
80/tcp   open  http
587/tcp  open  submission
3306/tcp open  mysql

Nmap finished: 1 IP address (1 host up) scanned in 1.009 seconds
```

Izpis 6-16: Primer iskanja odprtih vrat s SYN metodo

Dnevnik zajete aktivnosti iskanja odprtih vrat s SYN metodo:

```
=================================================================
 Data dump log:

 Created: 2009/06/14 13:48:05
 Remote IP: 127.0.0.1
 Server version: Simx Server 1.0.25 "PROTOTYPE"
 Drone version: Simx Driver 1.0.25 "PROTOTYPE"
=================================================================
[2009/14/06 13:46:47] portscan from 192.168.1.13:47754 -> 1410, type (0x4) Syn
[2009/14/06 13:46:47] portscan from 192.168.1.13:47754 -> 32777, type (0x4) Syn
[2009/14/06 13:46:47] portscan from 192.168.1.13:47754 -> 27374, type (0x4) Syn
[2009/14/06 13:46:47] portscan from 192.168.1.13:47754 -> 13783, type (0x4) Syn
[2009/14/06 13:46:47] portscan from 192.168.1.13:47754 -> 1447, type (0x4) Syn
[2009/14/06 13:46:47] portscan from 192.168.1.13:47754 -> 824, type (0x4) Syn
[2009/14/06 13:46:47] portscan from 192.168.1.13:47754 -> 238, type (0x4) Syn
```

Izpis 6-17: Dnevnik zajete aktivnosti iskanja odprtih vrat s SYN metodo

Simulacija prve faze napada – poizvedovanja. Primer iskanja odprtih vrat z nmap [vii] orodjem. Uporabljena je SYN metoda:

```
# nmap -sN node3.site-xyz.com

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2009-06-14
14:10 GMT+1
Interesting ports on localhost.localdomain (127.0.0.1):
Not shown: 1672 closed ports
PORT     STATE         SERVICE
21/tcp   open|filtered ftp
22/tcp   open|filtered ssh
23/tcp   open|filtered telnet
25/tcp   open|filtered smtp
79/tcp   open|filtered finger
80/tcp   open|filtered http
587/tcp  open|filtered submission
3306/tcp open|filtered mysql

Nmap finished: 1 IP address (1 host up) scanned in 2.177 seconds
```

Izpis 6-18: Dnevnik zajete aktivnosti iskanja odprtih vrat s SYN metodo

Dnevnik zajete aktivnosti iskanja odprtih vrat z NULL metodo:

```
=================================================================
 Data dump log:
```

```
 Created: 2009/06/14 14:19:17
 Remote IP: 127.0.0.1
 Server version: Simx Server 1.0.25 "PROTOTYPE"
 Drone version: Simx Driver 1.0.25 "PROTOTYPE"
================================================================

[2009/14/06 14:09:56] portscan from 192.168.1.13:35634 -> 3389, type (0x1) NULL
[2009/14/06 14:09:56] portscan from 192.168.1.13:35634 -> 21, type (0x1) NULL
[2009/14/06 14:09:56] portscan from 192.168.1.13:35634 -> 443, type (0x1) NULL
[2009/14/06 14:09:56] portscan from 192.168.1.13:35634 -> 22, type (0x1) NULL
[2009/14/06 14:09:56] portscan from 192.168.1.13:35634 -> 25, type (0x1) NULL
[2009/14/06 14:09:56] portscan from 192.168.1.13:35634 -> 80, type (0x1) NULL
[2009/14/06 14:09:56] portscan from 192.168.1.13:35634 -> 23, type (0x1) NULL
[2009/14/06 14:09:56] portscan from 192.168.1.13:35634 -> 389, type (0x1) NULL
[2009/14/06 14:09:56] portscan from 192.168.1.13:35634 -> 113, type (0x1) NULL
[2009/14/06 14:09:56] portscan from 192.168.1.13:35634 -> 554, type (0x1) NULL
[2009/14/06 14:09:56] portscan from 192.168.1.13:35634 -> 53, type (0x1) NULL
```

Izpis 6-19: Dnevnik zajete aktivnosti iskanja odprti vrat z NULL metodo

6.2.5.2 Dnevnik mrežnega napada

Simulacija napada je bila izvršena s pomočjo orodja Metasploit [27]. Za tip napada sem si izbral izkoriščanje preplavitve pomnilnika, na katero je ranljiv samba servis različic od 2.2.0 do 2.2.8 - Samba trans2open Overflow [26].

Konkreten opis simulacije napada:

```
$ ./msfconsole
+ -- --=[ msfconsole v2.8-dev [158 exploits - 76 payloads]

msf > use samba_trans2open
msf samba_trans2open >

msf samba_trans2open > set payload linux_ia32_bind
payload -> linux_ia32_bind

msf samba_trans2open(linux_ia32_bind) > set RHOST node2.site-xyz.com
RHOST -> node2.site-xyz.com

msf samba_trans2open(linux_ia32_bind) > check
[*] Target seems to running vulnerable version: Samba 2.2.0

msf samba_trans2open(linux_ia32_bind) > exploit
[*] WARNING: the correct case of the 'target' variable is 'TARGET'
[*] Starting Bind Handler.
[*] Starting bruteforce mode for target Linux x86
[*] Trying return address 0xbffffdfc...
[*] Trying return address 0xbffffbfc...
[*] Trying return address 0xbffff9fc...
[*] Got connection from 193.77.186.91:46954 <-> 89.212.17.80:4444

Id
uid=0(root) gid=0(root) euid=65534(nobody) egid=65534(nogroup)
groups=65534(nogroup)

w
 14:50:46 up 13 days,  4:01,  9 users,  load average: 0.06, 0.03, 0.02
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
simonm   pts/0    192.168.1.11     04Jun09  2:36m  1:26   0.21s -bash
simonm   pts/1    192.168.1.11     07Jun09 57:46   3.74s  3.34s ssh 0
simonm   pts/2    192.168.1.11     07Jun09  6:44   0.46s  0.46s -bash
simonm   pts/3    localhost.locald 07Jun09 57:46   0.29s  0.29s -bash
simonm   pts/5    localhost.locald 07Jun09  9days  1.67s  1.61s ssh 0
simonm   pts/6    localhost.locald 07Jun09  9days  0.30s  0.30s -bash
wwwadmin pts/7    192.168.1.11     12:43   57:32   0.02s  0.02s -bash
simonm   pts/8    192.168.1.100    12:55    0.00s  1.61s  1.54s
/usr/bin/perl .
```

```
unset HISTFILE
unset HISTSAVE
unset HISTLOG
```

Izpis 6-20: Konkreten opis simulacije napada

Z orodjem Snort je zajet dnevnik zaznanega napada, kjer je razviden tip napada:

```
[**] [1:2103:9] NETBIOS SMB trans2open buffer overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
07/18-12:30:37.816057 193.77.186.91:47938 -> 194.249.238.134:139
TCP TTL:53 TOS:0x0 ID:29658 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x476AC1E0 Ack: 0x2E249707  Win: 0xB68 TcpLen: 32
TCP Options (3) => NOP NOP TS: 214020820 6062617
[Xref=> <a href="http://www.digitaldefense.net/labs/advisories/DDI-
1013.txt">http://www.digitaldefense.net/labs/advisories/DDI-
1013.txt</a>]
[Xref => <a href="http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-
0201">http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0201</a>]
[Xref=> <a
href="http://www.securityfocus.com/bid/7294">http://www.securityfocus.c
om/bid/7294</a>]

[**] [1:648:7] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
07/18-12:30:37.817422 193.77.186.91:47938 -&gt; 194.249.238.134:139
TCP TTL:53 TOS:0x0 ID:29659 IpLen:20 DgmLen:1500 DF
***AP*** Seq: 0x476AC788 Ack: 0x2E249707  Win: 0xB68 TcpLen: 32
TCP Options (3) =&gt; NOP NOP TS: 214020820 6062617
[Xref =&gt; <a
href="http://www.whitehats.com/info/IDS181">http://www.whitehats.com/in
fo/IDS181</a>]
```

Izpis 6-21: Z orodjem Snort je zajet dnevnik zaznanega napada

Z orodjem Snort je zajet dnevnik rezultata napada, ki nam pove, da je napadalec izvršil zlonamerno kodo z nadzorniškimi pravicami:

```
[**] [1:498:6] ATTACK-RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
07/18-12:30:55.511182 10.2.2.120:45295 -> 69.44.XXX.XXX:48283
TCP TTL:64 TOS:0x0 ID:56468 IpLen:20 DgmLen:140 DF
***AP*** Seq: 0x2E1F5FBE Ack: 0x47D426D5  Win: 0x16A0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 6064395 214022589
```

Izpis 6-22: Z orodjem Snort je zajet dnevnik rezultata napada

6.2.5.3 Analiza napadalčeve aktivnosti na sistemu z muholovcem

- Dnevnik zajete aktivnosti na terminalih, kjer imamo nazoren vpogled napadalčevih ukazov, izvršenih na sistemu takoj po vdoru:

```
=================================================================
 SIMX Data dump log:
  Remote IP: 193.77.168.91
  Remote hostname: node2.site-xyz.com
  Remote OS: Linux (2.4.78,#2 on i386)
  Drone version: simx drone 1.0.7
  Create time: 2009/05/31 13:46:50
=================================================================

[2009/31/05 13:15:29 0 0] [UP][UP][UP]
[2009/31/05 13:15:38 0 0] id
[2009/31/05 13:16:09 0 0]
[2009/31/05 13:16:14 0 0] [BS]
[2009/31/05 13:16:16 0 0] w
[2009/31/05 13:16:19 0 0] dmesh[BS]g
```

```
[2009/31/05 13:16:59 0 0]
[2009/31/05 13:17:01 0 0]
[2009/31/05 13:17:01 0 0] unset HISTFILE
[2009/31/05 13:17:01 0 0] cat /etc/passwd
```

Izpis 6-23: Dnevnik zajete aktivnosti na terminalih

- Dnevnik zajete aktivnosti na datotečnem sistemu, kjer lahko vidimo, katere (in kako) datoteke so bile uporabljene med napadalčevo prisotnostjo na opazovanem sistemu:

```
========================================================================
SIMX Data dump log:
  Remote IP: 193.77.168.91
  Remote hostname: node2.site-xyz.com
  Remote OS: Linux (2.4.78,#2 on i386)
  Drone version: simx drone 1.0.7
  Create time: 2009/05/31 13:46:50
========================================================================

 [2009/31/05 13:15:38 0 0] open: /etc/group
[2009/31/05 13:16:09 0 0] open: /etc/passwd
[2009/31/05 13:16:14 0 0] open: /var/log/wtmp
[2009/31/05 13:16:16 0 0] open: /var/log/lastlog
[2009/31/05 13:16:59 0 0] open: /etc/ld.so.cache
[2009/31/05 13:17:01 0 0] open: /etc/passwd
[2009/31/05 13:17:02 0 0] read: /etc/passwd
```

Izpis 6-24: Dnevnik zajete aktivnosti na datotečnem sistemu

7 SKLEPNE UGOTOVITVE

"Information is power." je znan citat neznanega avtorja (vsaj meni ga ni uspelo najti), kateri še kako drži v boju z mrežnimi vdiralci. Namreč bolj kot poznamo sovražnika, večja je verjetnost, da se uspešno zaščitimo. Kar pa v realnosti dostikrat ne drži, saj so napadalčeve taktike in motivi pogostokrat ignorirani iz strani strokovnjakov za računalniško varnost. Omrežja z muholovec orodji pa so zasnovana ravno v te namene.

V okviru diplomske naloge sem se odločil v celoti razviti svojo verzijo muholovec orodja, z namenom spoznati uporabljene tehnike in koncepte v detajle, za kar mi je bilo med razvijanjem lastnega orodja vsekakor zagotovljeno. Po pregledu obstoječih orodij sem se osredotočil na danes najbolj razširjeno muholovec orodje Sebek [33], razvito v okviru organizacije Honeypot Project [33] in katerega izvorna koda je tudi prosto dosegljiva širši javnosti. Med razvojem orodja sem povzel veliko že znanih idej in konceptov, bodisi iz omenjenega muholovca Sebek ali iz drugih virov, najdenih na spletu, jih poizkušal izboljšati in združiti v zaokroženo celoto, katera bi lahko bila osnova za nadaljevanje raziskovalnega dela na to temo. Samo za primerjavo z orodjem Sebek naj omenim, da razvito orodje uporablja dosti bolj tesno integrirane metode interakcije z gostujočim operacijskim sistemom (prisotnost, prisluškovanje in mrežna aktivnost), kar nam posledično nudi manjšo verjetnost biti odkrit s strani napadalca. Pri komunikaciji z nadzorno točko in prenosu zajetega materiala ni več omejitve na samo UDP, temveč imamo na voljo še TCP in ICMP, kar nam nudi večjo prilagodljivost na uporabljeno okolje. Poleg tega sem se trudil zasnovati ogrodje orodja čim bolj modularno in pregledno, tako da je morebitno dodajanje nove funkcionalnosti preprostejše. Vendar je potrebno poudariti, da je razvito orodje še vedno v prototipni fazi in kot tako še neprimerno uporabi v produkcijske namene.

Za predstavitev praktičnega dela sem v okviru naloge postavil testni poligon, sestavljen iz več geografsko ločenih opazovalnih točk, katere sem navzven predstavil kot del porazdeljenega omrežja spletne strani, postavljene za vabo. Njen namen je generirati organski, neavtomatizirani mrežni promet, v kontekstu katerega je večja verjetnost, da pride do mrežnega napada. Vse opazovane točke sem opremil z različnimi vrstami senzorjev za podrobno opazovanje dogajanja: na vse točke sem namestil običajna orodja za odkrivanje mrežnih napadov, na (namenoma) najbolj izpostavljeno pa sem namestil še lastnoročno razvit (v namene diplomske naloge) visoko interaktivni muholovec, z namenom zajeti napadalčevo aktivnost po uspešnem vdoru v opazovani sistem. To bi lepo zaokrožilo rezultate, predstavljene v praktičnem delu naloge, vendar do dejanskega vdora v času opazovanja testnega poligona na žalost ni prišlo. Zato sem se odločil za predstavitev delovanja razvitega muholovca s simulacijo napada na opazovani sistem in predstavitvijo delovanja z zajemom simulirane aktivnosti. Poleg tega sem predstavil še odkrite vzorce napadov na vse opazovane točke: najbolj pogosti vzorci so različni poizkusi izkoriščanja ranljivosti spletnih aplikacij, nameščenih na spletni strani, pa vse do poskusov preplavitve pomnilnika, z namenom izvajanja zlonamerne kode ter neuspeli dostopi do zaščitenih datotek ...

Za zaključek lahko povzamem, da je, sodeč po podatkih, zbranih v času opazovanja aktivnosti na testnem poligonu, velika večina odkritih vzorcev napadov najverjetneje generirana s strani avtomatiziranih orodij, kot so naprimer mrežni črvi ali sorodna zlonamerna koda, za kar nam uporaba muholovca ne pomaga ravno najbolje. Vendar

sem prepričan, da bi ob nadaljnjem opazovanju prišli do konkretnejših rezultatov, zbranih ob dejanskem vdoru v opazovan računalniški sistem. Kajti ob prebiranju razultatov raziskovalnih nalog [34] na temo muholovcev sem zasledil, da so pred nakaj leti prišli do zaključkov, da je bil izmerjen čas od postavitve operacijskega sistema Windows 98 (navadna nastavitev z datotekami v skupni rabi, brez posodobitev) pa do uspešnega vdora, manj kot 24 ur! V primeru Linux (Red Hat) operacijskega sistema (zopet navadna postavitev, brez posodobitev) pa je najmanjši izmerjen čas od namestitve pa do uspešnega vdora bil manj kot 3 dni. Moji rezultati odstopajo od teh meritev, za kar vidim nekako dva razloga, namreč od omenjenih meritev je minilo že nekaj časa (let) in ker je v računalniškem svetu za spremembo trenov dovolj že manj kot eno leto ostaja velika verjetnost, da uporabljen testni poligon pač ni več v ciljni skupini (večinoma Microsoft Windows namizja) modernejših različic avtomatiziranih napadov. Preostanejo nam torej še neposredni napadi, do katerih pa sklepam da ni prišlo zaradi premajhne »vidljivosti« opazovanih kotrolnih točk. Z namenom povečati vidljivost sem testni poligon zasnoval kot spletno stran porazdeljeno na tri geografsko ločene točke, vendar tudi to ni bilo dovolj.

Med predloge za morebitno nadaljevanje raziskovalne naloge spadajo že omenjeno nadaljevanje opazovanje postavljenega poligona, razširitev le tega z dodatnimi kontrolnimi točkami (lahko v kontekstu spletne strani ali kaj drugega, potencialnemu napadalcu še bolj mamljivega) ter implementacija razvitega muholovca na drugih platformah, kot so Solaris, HP-UX in Microsoft Windows... še posebej slednja bi bila zanimiva za opazovanje aktivnosti mrežnih črvov. Poleg tega bi razviti muholovec, kateri je zaenkrat še v prototipni fazi, lahko ob nadaljevanju raziskovalnega dela »dozorel« in tako dosegel produkcijski nivo in bil kot tak primeren aktivni uporabi v kontekstu raziskovalne organizacije Honeypot Project [25].

8 PRILOGE

8.1 SLIKE

8.2 IZPISI

9 VIRI IN LITERATURA

[1] Addison-Wesley , *Know Your Enemy (2nd Ed.)*, ISBN 0-321-16646-9.

[2] *LaBrea* - Sticky honeypot. Dostopno na: http://labrea.sourceforge.net/

[3] *BackOfficer Friendl*" - Low interaction honeypot. Dostopno na: http://www.securityfocus.com/tools/2222

[4] *Spected* - Easy to deploy low interaction honeypot. Dostopno na: http://www.specter.com/

[5] *The Deception Toolkit*. Dostopno na: http://all.net/dtk/download.html

[6] *Snort* - The de facto standard for intrusion detection/prevention. Dostopno na: http://www.snort.org/

[7] IETF, *RFC 792 - Internet Control Message Protocol*. Dostopno na: http://www.faqs.org/rfcs/rfc792.html

[8] Clifford Stoll, *The Coocoo's egg*, ISBN 978-0743411462

[9] Lance Spitzer, *Honeypots: Tracking hackers*, ISBN 978-0321108951

[10] *IETF, "RFC 768 – User Datagram Protocol"* http://www.faqs.org/rfcs/rfc768.html

[11] IEEE projekt 802 http://en.wikipedia.org/wiki/IEEE_802

[12] Securityfocus , *MasterCard warns of massive credit-card breach.* Dostopno na: http://www.securityfocus.com/news/11219

[13] *Session Hijacking Exploiting TCP, UDP and HTTP Sessions,* Dostopno na: http://www.infosecwriters.com/text_resources/pdf/SKapoor_SessionHijacking.pdf

[14] IEFT, "RFC 793 – Transport Control Protocol". Dostopno na: http://www.faqs.org/rfcs/rfc793.html

[15] "Prenos DNS območja (DNS zone transfer)«, odstavek "Elesov preskok v spletno okolje". Dostopno na: http://tadejbogataj.110mb.com/datoteke/pdf/dns.pdf

[16] F. Cohen, *Computers and Security,* ISBN 0167-4048 http://portal.acm.org/citation.cfm?id=25139&CFID=25830171

[17] S.M. Bellovin, *Packets Found on an Internet*, ISSN 0146-4833. Dostopno na: http://portal.acm.org/citation.cfm?id=174199

[18] IETF, *RFC 2474 – Definition of the Diferentiated Services filed*. Dostopno na: http://tools.ietf.org/html/rfc2474

[19] IEFT, *RFC 3168 – The Addition of Explicit Congestion Notification (ECN) to IP*. Dostopno na: http://tools.ietf.org/html/rfc3168

[20] IETF, *RFC 791 – Internet Protocol*. Dostopno na: http://tools.ietf.org/html/rfc791

[21] *IETF, "RFC 790 - Assigned numbers.* Dostopno na: http://tools.ietf.org/html/rfc790

[22] Phrack, *Hacking the Linux Kernel Network Stack,* Volume 0x0b, Issue 0x3d. Dostopno na: http://www.phrack.com/issues.html?issue=61&id=13

[23] *THC-vlogger* - an advanced linux kernel based keylogger, http://linux.wareseeker.com/System/thc-vlogger-2.1.1.zip/332763

[24] Phrack, *Infecting loadable kernel modules,* Volume 0x0b, Issue 0x3d. Dostopno na: http://www.phrack.com/issues.html?issue=61&id=10

[25] *The Honeynet project* - A community of organizations actively researching, developing and deploying Honeynets and sharing the lessons learned. Dostopno na: http://www.honeynet.org/

[26] *The Metasploit Project* - The *Metasploit* Project is an open source computer security project which provides information about security vulnerabilities and aids in penetration testing. Dostopno na: http://www.metasploit.com/

[27] *Samba trans2open Overflow* - This exploits the buffer overflow found in Samba versions 2.2.0 to 2.2.8. This particular module is capable of exploiting the bug on x86 Linux and FreeBSD systems. Dostopno na: http://www.digitaldefense.net/labs/advisories/DDI-1013.txt

[28] Susan Young, Dave Aitel , *The Hacker's Handbook: The Strategy Behind Breaking into and Defending Networks*, Tools for Deletion or Update of Audit Files. Dostopno na: http://www.amazon.com/Hackers-Handbook-Strategy-Breaking-Defending/dp/0849308887

[29] *Rootkit* - Rootkits are powerful tools to compromise computer systems without detection. Dostopno na: http://www.rootkit.com

[30] Remote shell over ICMP. Dostopno na: http://www.phrack.org/archives/51/P51-06

[31] CVE-2002-0649, *Multiple buffer overflows in the Resolution Service for Microsoft SQL Server 2000.* Dostopno na: http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0649

[32] *SQL slammer (computer worm)*. Dostopno na: http://en.wikipedia.org/wiki/SQL_slammer_(computer_worm)

[33] *Sebek* - kernel module installed on high-interaction honeypots for the purpose of extensive data collection. Dostopno na: http://www.honeynet.org/tools/sebek/

[34] The Honeypot Project, *Know Your Enemy: Statistics - Analyzing the past ... predicting the future.* Dostopno na: http://www.noncombatant.org/trove/honeynet-papers/stats/index.html.org

10 ORODJA

[i] Jeremy Chartier <jeremy.chartier@free.fr>, *snortalog.pl* - SnortALog is a powerfull perl script that summarizes snort logs.

[ii] *Snort* - the de facto standard for intrusion detection/prevention. Dostopno na: http://www.snort.org/

[iii] *Nessus* - Vulnerability Scanner from Tenable Network Security. Dostopno na: http://www.nessus.org

[iv] *TripWire* - Tripwire's powerful configuration assessment and change auditing solutions let IT gain configuration control of the entire IT infrastructure. Dostopno na: http://www.tripwire.com/

[v] *simx* – High interaction honeypot solution. Dostopno na: http://git.site-xyz.com/git/simx/

[vi] *site-xyz* - spletna stran, postavljena kot vaba katere namen je generirati organski (neavtomatiziran) mrežni promet v kontektsu katerega bi eventuelno lahko prišlo do poizkusov mrežnega napada. Dostopno na: http://www.site-xyz.com