Implementacija i konfiguracija Google Rapid Response forenzičkog alata u svrhu udaljenog otkrivanja, nadzora i uklanjanja štetnog programskog koda

Komparić, Aleksandar

Master's thesis / Specijalistički diplomski stručni

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Algebra University College / Visoko učilište Algebra**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:225:636099

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-18**

Repository / Repozitorij:

Algebra Univerity College - Repository of Algebra Univerity College

VISOKO UČILIŠTE ALGEBRA
VISOKA ŠKOLA ZA PRIMIJENJENO RAČUNARSTVO

DIPLOMSKI RAD

Implementacija i konfiguracija Google Rapid Response forenzičkog alata u svrhu udaljenog otkrivanja, nadzora i uklanjanja štetnog programskog koda

Aleksandar Komparić

Zagreb, svibanj 2017.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Ovaj rad daje uvid u principe rada Google Rapid Response alata za udaljenu forenziku i odgovor na incidente. Opisuje elemente koje alat koristi te daje pregled mogućnosti koje alat posjeduje. Alat je implementiran u testnu okolinu te su pokrenuti moduli zvani tokovi i lovovi koji su ocijenjeni zanimljivima za analizu i potencijalnu detekciju malicioznih programa ili korisnika, po procijeni autora. Drugi dio rada se sastoji od dokumentiranja korištenja GRR-a, nailaženja na greške te pokušaje ispravljanja istih. GRR je ocijenjen kao obećavajući, ali ne i jednostavan za primjenu i korištenje u trenutnom izdanju, primarno zbog nekonzistentnosti u radu, velikog broja grešaka na minimalno kompleksnoj infrastrukturi te lošije izvedenog sučelja koje ne prikazuje bitne informacije na pregledan način.

Ključne riječi: Google Rapid Response, računalna forenzika, zaštita, odgovor na incidente

Summary

This thesis provides an insight into the principles of the Google Rapid Response framework, a tool for remote forensics and incident response. It describes the elements of the tool and gives an overview of the features the tool has at its disposal. The application has been implemented in a test environment in which certain modules called hunts and flows have been executed based on their level of usefulness for analysis and detection of malicious applications and users, as perceived by the author. The second part of the thesis describes the usage of GRR, its errors and authors attempts to troubleshoot them. GRR is labelled as a promising tool that is not easy to implement or manage in its current form, primarily because of the inconsistencies in operation and a large number of errors in an environment of a very limited complexity, as well as poorly designed interface that does not bring important information into focus in a visually clear way.

Key words: Google Rapid Response, computer forensics, incident response

Sadržaj

1. Uvod 1

2. Osnovno o Google Rapid Responseu 2

2.1. Princip rada 2

2.2. Tok (*Flow*) 3

2.3. Lov (*Hunt*) 4

2.4. Isporuka i izvršavanje specijalnih pokretačkih programa i koda 5

2.5. Artefakti 6

2.6. AFF4 6

2.7. Rekall forenzika memorije 7

2.8. SleuthKit 7

2.9. Ostale mogućnosti 8

2.10. Budućnost 8

3. Opis testne okoline 10

4. Instalacija i konfiguracija 12

4.1. Server 12

4.2. Klijent 13

4.3. Web sučelje 13

4.4. Konzola 17

4.5. Predefinirani tokovi u sustavu 18

5. Izvođenje tokova 20

6. Izvođenje lova 23

6.1. Izvođenje lova po vremenskom rasporedu 25

7. Greške 27

7.1. Lovovi/tokovi 27

7.2. Pokušaji otklanjanja grešaka 33

8. Zaključak 35

Popis kratica 37

Popis slika 38

Literatura 39

1. Uvod

Današnji računalni sustavi sve su češće podložni napadima protiv kojih se klasični antivirusni alati ne mogu boriti. U takvim situacijama, gdje se preventiva svodi na odgovorno korištenje računalnih sustava, vrlo je bitno moći utvrditi uzrok, odnosno inicijalni vektor napada, kako se isti ne bi ponovio u budućnosti.

Implementacija nadzora svih radnih stanica u jednoj tvrtki nije jednostavna za izvesti i često se svodi na korištenje komercijalnog softvera koji predstavlja pozamašnu financijsku investiciju, ili korištenje besplatnih programa sa znanjem da je podrška za te programe upitna te da će se zadaću implementacije i održavanja sustava trebati dodijeliti nekom od (ili više) zaposlenika tvrtke.

Google Rapid Response Framework (GRR) spada u ovu drugu skupinu, no ima određene prednosti kojima se ističe nad konkurencijom.

Ovaj rad će dati opis rada GRR-a, koji će se primarno oslanjati na dokumentaciju objavljenu od strane tima koji ga je razvio. Ovaj opis je nužan da bi se razumio princip rada GRR-a te da kasnija poglavlja nisu razbijena opisima funkcionalnosti i modula GRR-a. Nakon opisa funkcionalnosti će se opisati okolina koja je složena za testiranje te potom vlastiti tok korištenja i konfiguracije softvera, počevši od serverske komponente, potom klijenata te na kraju konfiguracije cjelokupne okoline i praćenje rada svih sustava u okolini. Uzeti će se u obzir opterećenja klijentskih sustava prilikom izvršavanja kompleksnijih radnje te kolika je brzina i točnost detekcije kada se na klijente pusti maliciozni programi. Mjerenje opterećenja serverske komponente nije dio ovog rada, jer bi za korisne metrike bio potreban znatno veći broj sustava od onih koji će biti korišteni u ovom testu.

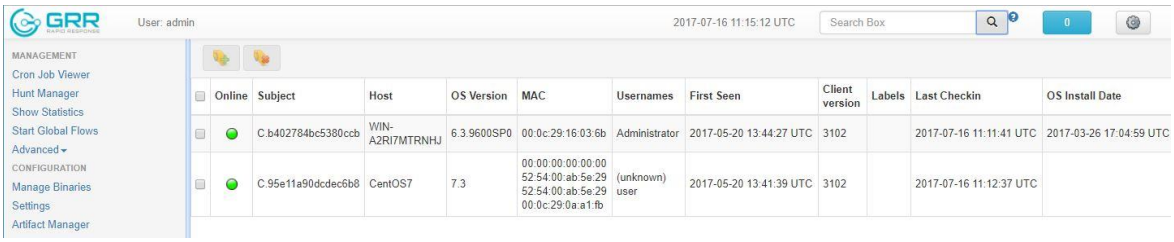
Svrha rada je primarno opisati GRR i njegove posebnosti te ustanoviti njegovu korisnost kao alata za praćenje heterogene infrastrukture u svrhu brze zaštite od nepoželjnih ili malicioznih radnji. Na kraju će biti dane opaske i procjena korisnosti korištenja alata u odnosu na težinu instalacije, konfiguracije i održavanja.

2. Osnovno o Google Rapid Responseu

2.1. Princip rada

Google Rapid Response je alat za udaljenu forenzičku analizu. Alat se dijeli na serversku i klijentsku komponentu, gdje više klijenata može kontaktirati jedan server. Na taj način je moguće imati centralnu upravljačku ploču za sve agente u okolini. Serverska i klijentska aplikacija su napisane u programskom jeziku Python, te su potpuno otvorenog koda koji je dostupan na GitHub web repozitoriju. Zahvaljujući tome, osobe sa adekvatnim znanjem Python programskog jezika mogu potpomoći razvoju ili prilagoditi određene mogućnosti bilo klijentske ili serverske aplikacije vlastitim potrebama.

Višestruki klijenti kontaktiraju server kojemu prijavljuju svoj status i osnovne informacije o sustavu na koji su instalirani.



Slika 1 GRR konzola - Pregled klijenata

Kao što je vidljivo sa Slika 1 GRR konzola - Pregled klijenata, informacije koje klijenti automatski sakupljaju i prijavljuju su status (je li klijent aktivan i trenutno dostupan za komunikaciju sa serverom), Subject (interna jedinstvena oznaka klijenta u GRR okolini), ime sustava na koji je klijent instaliran, inačica operacijskog sustava, fizička adresa, odnosno adrese sustava, korisnička imena koja imaju pravo prijave na sustav, vrijeme kada se klijent prvi puta javio serveru, opcionalne oznake kojima se olakšava upravljanje većim brojem sustava, vrijeme kada se klijent zadnji puta javio serveru te vrijeme instalacije operacijskog sustava, ako se može iščitati.

Jednom kad se jave serveru, klijenti su spremni za izvršavanje zadataka koje im zada server. Komunikacija između klijenata i servera može biti enkriptirana, omogućavajući primjenu preko interneta, što je pogodno organizacijama koje imaju decentraliziranu radnu okolinu. Naravno, kod takvih primjena treba uzeti u obzir obim posla koji se izvodi i čiji se rezultati trebaju poslati preko vanjskog linka da bi se izbjeglo zagušenje istog. (Castle, et al., 2017)

Serverska komponenta službeno podržava instalaciju samo na 64 bitni Ubuntu Xenial serverski operacijski sustav. Ranija inačica GRR-a je prešla na isključivo systemd init sistem te sada službeno podržava isključivo njega. Premda nije podržano, GRR se može instalirati i na ostale Linux serverske operacijske sustave, no tada su potrebne modifikacije servisa koje koristi GRR serverska komponenta. U ovom radu se koristio Ubuntu Xenial kao operacijski sustav za serversku komponentu.

Prilikom postavljanja servera GRR automatski generira klijentske instalacijske programe koji imaju već ugrađene postavke za komunikaciju sa serverom, na taj način omogućavajući najjednostavniju moguću instalaciju klijenata koja također može biti vrlo lako automatizirana za instalaciju na višestruke Linux klijente ili kroz Windows objekte grupne politike (*Group Policy Objects, GPO*). Jednom na sustavu, klijenti rade u pozadini čekajući na naredbe sa servera uz periodično slanje statusa serveru. Pored glavnog klijentskog procesa, na sustavu je također aktivan i zasebni proces „dadilja“(*nanny*) čija je uloga nadziranje rada klijentskog procesa te njegovo terminiranje ako isti probije definirane granice trajanja izvođenja procesa ili zauzeća procesora, odnosno radne memorije. (Castle, et al., GRR Administrator Documentation, 2017)

2.2. Tok (*Flow*)

Uobičajena analiza klijenta podrazumijeva poduzimanje određenog broja akcija koje se moraju izvršiti slijedno kako bi postigle tražene rezultate. Neefikasno je da server koji izdaje akcije stalno čeka odgovor klijenta, jer to zahtijeva zauzeće određenih resursa na serveru dok ne dobije odgovor. GRR koristi takozvane tokove (*flows*). Tok je model izvođenja koji dozvoljava obustavljanje akcija. To znači da se na serveru mogu zadati akcije klijentu, potom obustaviti izvođenje tog procesa sve dok klijent ne javi da je tražena akcija izvedena te preda rezultate. Nakon toga server se vraća u svoj proces te pokreće slijedeću akciju na klijentu. Procesi na serveru se mogu obustaviti na neograničen period vremena bez trošenja serverskih resursa. Korištenje tokova najviše dolazi do izražaja u većim okruženjima, gdje veliki broj klijenata stalno izvršava određene akcije, neki prolaze kroz periode nedostupnosti, dok se drugi uklanjaju jer se više ne koriste.

Tokovi omogućavaju da server uvijek ima slobodne resurse na raspolaganju, što rezultira brzim izdavanjem novih naredbi i neprekinutim primanjem podataka sa klijenata. Kad server pokrene tok na klijentu, on pošalje tražene instrukcije klijentu dok ne dođe u stanje u kojem

čeka na rezultate od klijenta. Nakon toga server otpušta sve resurse koje je taj tok držao, zapisuje unos u bazu, te čeka sve dok ne dobije odgovor od klijenta. Kad klijent pošalje odgovor, server ponovno dohvaća sve nužne resurse za tok i pokreće tok u stanju gdje je prethodno stao i procesira odgovor klijenta. Ako tok zahtijeva dodatne akcije, ova procedura se ponavlja, u suprotnom se rezultati toka zapisuju i on se zatvara.

Jednostavni tokovi, oni kojima je cijela logika akcije na serverskoj strani i nemaju dodatna stanja na klijentu, mogu perzistirati i kroz ponovno pokretanje klijentskog stroja. Naime, kako je server taj koji sadrži sve informacije i instrukcije o toku, klijent će uvijek pitati server koje akcije treba izvesti te će server uvijek poslati naredbe koje klijent treba izvršiti sve dok klijent ne pošalje traženi odgovor.

Pošto se višestruki tokovi mogu izvoditi na klijentu u isto vrijeme, svaki tok ima svoj jedinstveni sesijski identifikator (*session ID*) pomoću kojega server može razaznati o kojem se toku radi kada mu klijent vrati odgovor. Jedini tokovi koji ne koriste taj jedinstveni identifikator su takozvani dobro znani tokovi (*Well Known Flows*). Ti tokovi imaju poznati identifikator koji ne zahtijeva inicijalnu komunikaciju sa serverom. Oni se koriste za akcije koje klijent mora inicirati (kao što je na primjer prvo javljanje serveru nakon instalacije klijenta ili pokretanje drugih tokova). Zbog prirode njihove komunikacije, dobro znani tokovi ne čuvaju stanje (*stateless*). (Castle, et al., GRR Administrator Documentation, 2017)

2.3. Lov (*Hunt*)

Lov je u suštini serija tokova koji se izvršavaju na većem broju klijenata u isto vrijeme. Pomoću njih se mogu tražiti artefakti malicioznih programa ili analizirati uzroci neuobičajenog ponašanja na više strojeva u okolini. Lovovi omogućavaju da u kratko vrijeme dobijemo traženu informaciju sa svih aktivnih strojeva u okolini, minimizirajući potrebno vrijeme za provjeru.

Parametri lova su slijedeći:

- Opis – Opis lova
- Limit broja klijenata – Maksimalni broj klijenata na kojima se lov može izvršavati
- Vrijeme isteka – Za klijente koji se jave serveru nakon definiranog vremena, lov se neće pokrenuti

- Stopa povećanja broja klijenata – Broj klijenata na kojima se lov pokreće po minuti. Cilj ovog parametra je da se ne preoptereti server kada se izvode lovovi koji šalju veći broj poruka serveru.

Lov je najlakše rasporediti na grupe, odnosno oznake (*labels*), gdje se isti potom primjeni na sve aktivne članove koji su dodijeljeni toj oznaci. Međutim, unutar samog lova se mogu odrediti razni parametri koji se mogu koristi za odabir klijenata (na primjer, verzija operacijskog sustava ili strojevi čije ime počinje sa istim predloškom).

Lov se može pokrenuti kroz web sučelje ili kroz konzolu, uobičajeno se web sučelje koristi za predefinirane lovove, dok bi se u konzoli napisao lov koji nije definiran ranije. Lov se kroz konzolu može spremiti kako bi se kasnije mogao koristiti i kroz web sučelje.

2.4. Isporuka i izvršavanje specijalnih pokretačkih programa i koda

Koristeći zadane vrijednosti, GRR zahtijeva posebne provjere prije pokretanja koda koji nije isporučen sa samim klijentom. Na taj način se smanjuje mogućnost iskorištavanja GRR klijenata za izvođenje malicioznih programa. Ova mogućnost je dodana jer prilikom forenzičke analize koji put treba koristiti specijalne programe ili pokretačke programe (*driver*) za koje se ne očekuje učestalo korištenje pa nema potrebe da se ugrađuju u samoga klijenta. Još jedna mogućnost koja koristi izvršavanje koda je ažuriranje samoga klijenta koristeći centralni GRR server, čime se znatno smanjuje vrijeme održavanja GRR infrastrukture.

Da bi klijent izvršio kod ili učitao traženi upravljački program, isti mora biti potpisan sa privatnim ključem za kod (*Client.executable_signing_public_key*) ili za upravljački program (*Client.driver_signing_public_key*). Klijent provjerava te ključeve te ako ih može potvrditi sa svojim ključem, kreće u izvršavanje dostavljenog koda. Kod se šalje klijentu kao protobuf (Google-ovo rješenje za serijalizaciju strukturiranih podataka koje je neutralno, odnosno neovisno o programskom jeziku i o platformi na koju se isporučuje), u kojem se pored isporučenog koda nalazi i kriptografski sažetak (*hash*) za provjeru ispravnosti pristiglog paketa, potpis za provjeru vjerodostojnosti paketa i dodatni konfiguracijski podaci, primarno vezani uz dodatne parametre za izvršavanje dostavljenog koda ili upravljačkog programa.

Pokretački programi moraju biti kompatibilni sa Rekall forenzičkim alatima. Kod mora biti pisan u Python programskom jeziku i funkcionirati kada je pokrenut sa *exec()* naredbom u sigurnosnom kontekstu GRR klijenta. Sve informacije koje se trebaju poslati na GRR server trebaju biti spremljene u *magic_return_str* varijablu.

Potpisivanje koda za dostavljanje se vrši korištenjem *config_updater* pomoćnog programa i ne mora se izvršiti na GRR serveru, ako bi se htjela povisiti razina sigurnosti. (Castle, et al., GRR Administrator Documentation, 2017)

2.5. Artefakti

U sklopu forenzičke istrage potrebno je prikupiti podatke koji se očekuju u sustavu, kao što su zapisi sustava, servisi koji se izvode, korisnički računi i slično. Ti podaci se u forenzičkoj istrazi nazivaju artefakti i njihova lokacija varira od sustava do sustava. Prikupljanje artefakata spada u jednostavnije aspekte korištenja GRR-a.

GRR je implementirao radni okvir za opisivanje forenzičkih artefakata koji omogućava da se oni što brže prikupe i kategoriziraju. Ciljevi tog radnog okvira su precizno opisivanje artefakata za jednostavnije definiranje automatskog prikupljanja od strane sustava, konfiguracija koja omogućava jednostavno definiranje objekata koje treba prikupiti, grupiranje aplikacijskih podataka sa raznih operacijskih sustava (moderne inačice operacijskih sustava Windows, Mac OS i Linux) te jednostavna definicija novih artefakata za njihovo što brže prikupljanje. (Castle, et al., GRR User manual, 2017)

2.6. AFF4

AFF4 su razvili Michael Cohen, Simson Garfinkel i Bradley Schatz. AFF4 je proširenje AFF1 forenzičkog datotečnog sustava. Razlog za razvijanje novog forenzičkog datotečnog formata je bio nezadovoljstvo postojećim formatima, bilo zbog pretjerane kompleksnosti (autori navode primjer EWF-a razvijenog od tvrtke Encase kao primjer datotečnog formata koji je kompleksan za implementaciju i korištenje), ili pretjerane jednostavnosti, odnosno nedostatka mogućnosti. Tu se kao primjer navodi dd, koji nativno ne podržava nikakav oblik kompresije i ne sprema metapodatke, digitalne potpise niti ima kriptografsku podršku. Ograničenja AFF1 datotečnog sustava su također nešto što su gore navedeni autori htjeli zaobići. Sa AFF4 se ukomponirala mogućnost spremanja više tokova podataka iz raznih izvora u jedan spremišni dokazni volumen. AFF1 sprema jednu sliku diska u svaku dokaznu

datoteku, što je suprotno metodi izvođenja modernih forenzičkih istraga, gdje jedan slučaj često uključuje veći broj računala i sredstava za pohranu podataka. AFF1 nije predviđen za pohranu preslike radne memorije ili uhvaćenih mrežnih paketa, niti je prilagođen za distribuiranu istragu, gdje više istražitelja istovremeno radi na jednom slučaju. Uobičajene metode označavanja objekata sastoje se od korištenja vremena prikupljanja dokaza sa kratkim opisima, što često dovodi do nekonzistentnih i potencijalno jednakih oznaka za različite objekte. Distribuirana istraga značajno povećava mogućnost korištenja identičnih identifikatora dokaznih objekata koji se prilikom objedinjavanja svog dokaznog materijala moraju promijeniti, zajedno sa svim potencijalnim vanjskim referencama. AFF4 uvodi globalno jedinstveni identifikator koji se automatski dodjeljuje prilikom uvođenja objekta u dokaznu datoteku, zvanu AFF4 URN (*Uniform Resource Name,* Ujednačeno ime resursa). Primjer jedne AFF4 URN oznake je *urn:aff4:bcc02ea5-eeb3-40ce-90cf-7315daf2505e*. (Cohen, Garfinkel, & Schatz, 2009)

2.7. Rekall forenzika memorije

GRR za forenziku radne memorije strojeva koristi Rekall alate za forenziku memorije. Rekall alati su dizajnirani da se izvode na računalu kojeg se analizira i to ih čini idealnima za primjenu u GRR. Oni omogućuju brzu analizu i mogu raditi korelaciju artefakata iz memorije sa onima koji se mogu prikupiti kroz sistemske pozive. Također, velika prednost je što se prikupljanje može raditi na računalu dok radi, bez potrebe za izvlačenjem cjelokupne memorije na drugi medij za pohranu u svrhu analize. To je poželjno na sustavima koji aktivnije koriste radnu memoriju jer znatno manje vremena prođe tijekom žive analize nego čekanjem na izvlačenje memorije na vanjski uređaj za analizu. GRR koristi Rekall pokretačke programe za operacijske sustave Windows, Linux i OS X. (Rekall Forensics , 2017)

2.8. SleuthKit

GRR server koristi *The Sleuth Kit (TSK)* za pristup i analizu sirovih datotečnih sustava operacijskih sustava. TSK je biblioteka u programskom jeziku C koju su razvili Brian Carrier, Dan Farmer i Wietse Venema za analizu datotečnih sustava bez ometanja rada operacijskog sustava. Zbog toga što ne ovise o operacijskom sustavu na kojem se izvršavaju, alati iz TSK mogu prikazati sadržaj koji ne bio vidljiv iz operacijskog sustava (npr. sakriveni

ili izbrisani sadržaj). TSK podržava NTFS, FAT, ExFAT, UFS 1, UFS 2, EXT2FS, EXT3FS, Ext4, HFS, ISO 9660 i YAFFS2 datotečne sustave, čak i na operacijskim sustavima koji ne podržavaju neke od njih. Još jedna prednost ne korištenja operacijskog sustava za datotečni pristup je što može prikazati datoteke raznih rootkit zloćudnih aplikacija koje bi bilo nemoguće vidjeti koristeći operacijski sustav na kojem se izvršavaju. (Carrier, 2017)

2.9. Ostale mogućnosti

GRR podržava uvoz i korištenje NSRL-a (*National Software Registry List*), skup poznatog softvera i pripadajućih datoteka koje se koriste, kako bi smanjio opseg forenzičke istrage tako da se preskoče datoteke čiji kriptografski sažetak odgovara onome u NSRL-u.

GRR koristi FUSE (*Fileystem in Userspace,* Datotečni sustav u korisničkom segmentu) sloj za pregled datotečnih sustava klijenta na server stroju. FUSE sloj omogućuje korisnicima kreiranje datotečnih sustava na stroju bez potrebe za modifikacijom kernela na način da ih stvara u korisničkom segmentu, dok FUSE sloj služi za premošćivanje kernel segmenta. Prilikom korištenja FUSE sloja, klijentima se automatski zadaju nužni tokovi sa kojima se prikupljaju nužni podaci za rad udaljenog datotečnog sustava. Ova funkcionalnost omogućuje brzi pregled udaljenog datotečnog sustava bez potrebe za kompleksnim komandama u svrhu brzog pregleda datoteka koje se u njemu nalaze. (Castle, et al., GRR Administrator Documentation, 2017)

2.10. Budućnost

Kako se GRR stalno razvija, nove mogućnosti se dodaju dok se neke stare zamjenjuju. Ovdje će se dati kratak pregled promjena koje se uvode u GRR sa prezentacije iz ožujka 2017.

Najveća promjena je izdvajanje komunikacije klijenata i servera u zasebnu komponentu zvanu Fleetspeak. Ona će se koristiti za efikasniju razmjenu poruka i identifikatora te u svrhu provjera prošlih akcija pošto Fleetspeak sprema komunikacije koje obradi, dok također smanjuje opterećenje sustava jer pokreće klijenta kad se dobije naredba koju treba izvršiti, što znači da klijent više ne treba biti stalno aktivan.

Očekuje se daljnje proširenje GRR API-a u svrhu još lakše automatizacije i integracije sa drugim sustavima. Međutim, podrška za novije verzije OS X-a i Linuxa upitne, razvojni tim će pokušati podržati nove Windows operativne sustave. Također se istražuje potencijalno

prebacivanje cjelokupnog GRR-a na Go programski jezik (Golang), a Fleetspeak komponenta je već napisana u tom programskom jeziku. (Moser, Bushkov, Galehouse, & Lakomy, 2017)

3. Opis testne okoline

Zbog praktičnih razloga, u sklopu testiranja korištena je virtualizacija kako bi se prikazao rad GRR-a na više različitih operacijskih sustava u paralelnom radu. Kao virtualizacijski sustav iskorišten je VMware Workstation 12.5, zbog toga što predstavlja prokušano i adekvatno rješenje za ovaj rad. U sklopu rada, napravljene su tri nove virtualne mašine, specificirane ispod:

1. UbuntuXenial:
    - OS: Ubuntu Xenial 16.04 64 bit
    - 2 virtualna procesora
    - 6 GB RAM-a
    - 120 GB alocirano za disk
2. Windows Server 2012:
    - OS: Windows Server 2012 r2 64 bit
    - 1 virtualni procesor
    - 2 GB RAM-a
    - 40 GB alocirano za disk
3. CentOS:
    - OS: CentOS 7.2 64 bit
    - 1 virtualni procesor
    - 2 GB RAM-a
    - 20 GB alocirano za disk

Nakon uočenih problema sa CentOS operacijskim sustavom, pripremljena je nova virtualna mašina sa Ubuntu operacijskim sustavom, čiji se detalji nalaze ispod:

- UbuntuClient
    - OS: Ubuntu LTS 16.04.3
    - 1 virtualni procesor
    - 2 GB RAM-a
    - 20 GB alocirano za disk

Na UbuntuXenial je instalirana serverska komponenta GRR-a. Windows Server 2012 i UbuntuClient, kao i CentOS prije njega, imaju instalirane GRR klijente koji se javljaju

serveru. Da bi ova komunikacija bila uspješna, modificirane su *hosts* datoteke svih klijentskih strojeva da sadrže DNS ime Ubuntu Xenial servera i njegovu IP adresu u svrhu omogućavanja komunikacije putem DNS imena. Pored ažuriranja i instalacije GRR klijenta, nikakve dodatne izmjene nisu inicijalno napravljene ovim sustavima. Svi sustavi koriste NAT mrežnu karticu dostupnu za korištenje u VMware alatu koja omogućava komunikaciju između strojeva, kao i sa internetom kroz sučelje računala na kojem se virtualne mašine izvode.

Instalirana i korištena inačica GRR-a je 3.1.0.2, zadnja dostupna gotova inačica u vrijeme pisanja rada.

4. Instalacija i konfiguracija

4.1. Server

Instalacija GRR serverske komponente je relativno jednostavna radnja, jer je većina koraka automatizirana skriptom koju se preuzme sa službenih repozitorija. Skripta se sama brine za instalaciju svih potrebnih paketa koji su nužni za rad, te vrši sve radnje za funkcionalno izvođenje servera. U slučaju da se ne koristi Ubuntu Xenial, skriptu bi trebalo prilagoditi operacijskom sustavu.

Jedna od odluka prilikom instalacije je odabir baze za pohranu podataka. GRR dolazi sa ugrađenom SQLite bazom, no preporuča se korištenje MySQL baze (inačice 5.6 ili novije) ako će se koristiti veći broj klijenata, odnosno za primjenu u veće okruženje. Prednost korištenja MySQL baze je u tome što se može nalaziti na drugom stroju time smanjujući opterećenje na jedan stroj, ali je u tom slučaju potreban brzi link između dva stroja. (Castle, et al., GRR Administrator Documentation, 2017)

Nakon toga se podešava ime GRR servera koje će biti korišteno za pristup administrativnoj konzoli i za komunikaciju sa klijentima. Potom slijedi URL adresa na koju će se klijenti pozivati prilikom komunikacije sa serverom, bitno je odabrati adresu koja će biti vidljiva klijentima kao i slobodan port na serveru. Zadani port je 8080, ali može se izmijeniti u ovom koraku. Kako se u ovom radu koristila virtualna mreža, u hosts datoteke klijenata je dodan unos koji IP adresu servera povezuje sa imenom koje je tu postavljeno. U produkcijskoj okolini bi se to napravilo korištenjem DNS servera. Slijedeće se postavlja URL adresa za administrativno sučelje servera. Zatim slijedi konfiguracija servera za elektroničku poštu koju će GRR koristiti za slanje poruka o stanju sustava. Definiraju se dvije email adrese, jedna na koju će se slati poruke vezane uz nadzor rada sustava kao što su rušenja klijenata ili servera. Druga email adresa se konfigurira za slanje hitnih i osjetljivih poruka, kao što je na primjer detektirano zaobilaženje ugrađenih kontrola.

Nakon ovih koraka postavlja se lozinka za pristup web sučelju admin korisnika. Na kraju, instalacijska skripta pokreće preuzimanje i konfiguraciju klijentskih programa sa predlošcima koje su definirani u prethodnim koracima. Sa time konfiguracija servera završava i on je dostupan putem web sučelja i konzole.

4.2. Klijent

Klijentske instalacije se preuzimaju sa servera, bilo kopiranjem iz repozitorija ili preuzimanjem putem web sučelja. GRR automatski kreira instalacijske pakete za operacijske sustave Windows (32 i 64 bitne), Linux (32 i 64 bitne, deb i rpm paketi) te za OS X. Klijentske instalacije su automatizirane i nema potrebe za unošenjem detalja prilikom instalacije, što je napravljeno namjerno kako bi automatska instalacija bila jednostavna za izvesti. Jednom instaliran klijent automatski kreće s radom i šalje serveru svoje podatke kako bi se prikazao u konzoli.

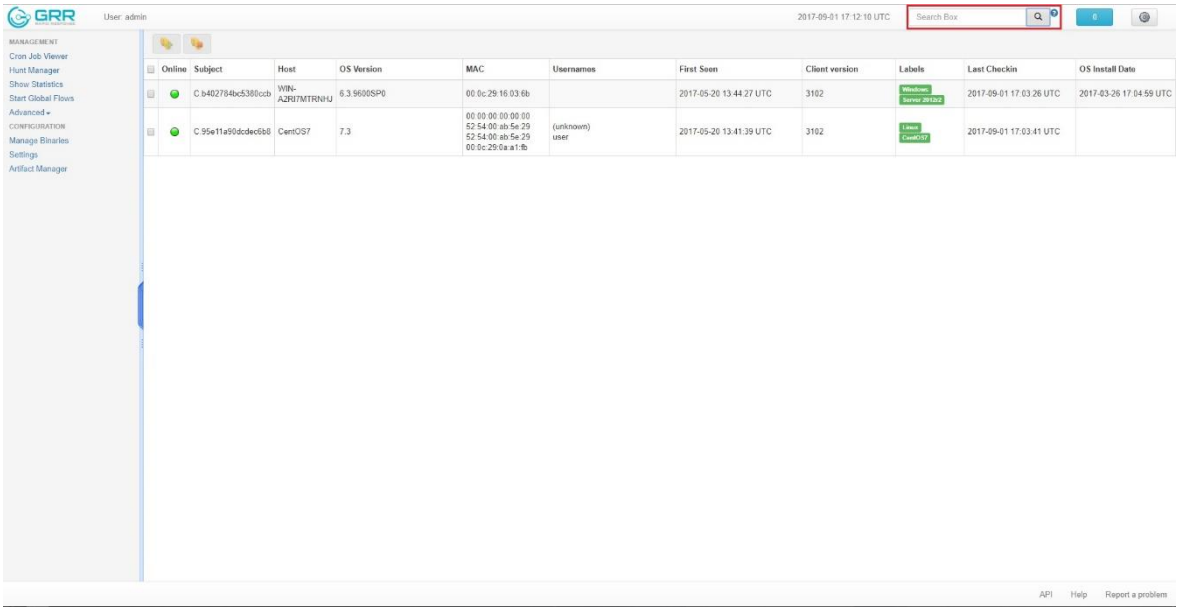
4.3. Web sučelje

Web sučelje GRR-a je dosta rudimentarno i ukazuje na nekomercijalnu prirodu proizvoda gdje prezentacija i jednostavno, odnosno intuitivno korisničko sučelje nisu elementi na koje su razvojni inženjeri softvera odlučili utrošiti vrijeme. U trenutnoj fazi proizvoda gdje se još uvijek dosta vremena i resursa koristi na razvijanje funkcionalnosti to je i za očekivati, ali osobno mišljenje autora je da se to neće previše promijeniti u budućnosti.

Web sučelje sa inicijalnom konfiguracijom dolazi isključivo sa basic autentikacijom. Web stranica traži korisničko ime i lozinku za pristup, ali promet nije enkriptiran što predstavlja problem u bilo kojoj poslovnoj okolini, pogotovo za sustav koji može skupljati osjetljive informacije sa strojeva. Za rješenje ovog problema razvijatelji softvera preporučuju konfiguraciju Apache reverse proxy-a na serveru koji će odrađivati proces autentikacije, zaštiti pristup web sučelju enkriptiranom vezom te također vršiti logiranje pristupa, odnosno pokušaja pristupa na web sučelje. (Castle, et al., GRR Administrator Documentation, 2017)

Slika 2 GRR Konzola

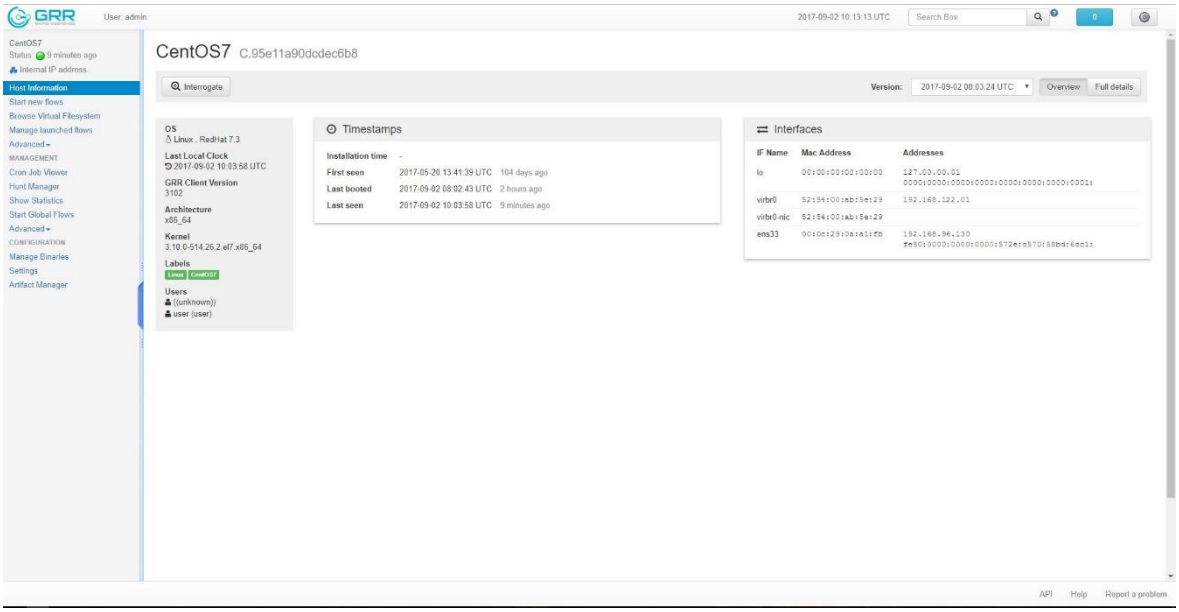
Glavna konzola (Slika 2 GRR Konzola) ne prikazuje puno informacija sve dok se ne selektira klijent. Premda je izvedeno nekoliko lovova i ostvaren pristup na oba klijenta, glavna konzola se nikad nije ažurirala sa tim podacima.



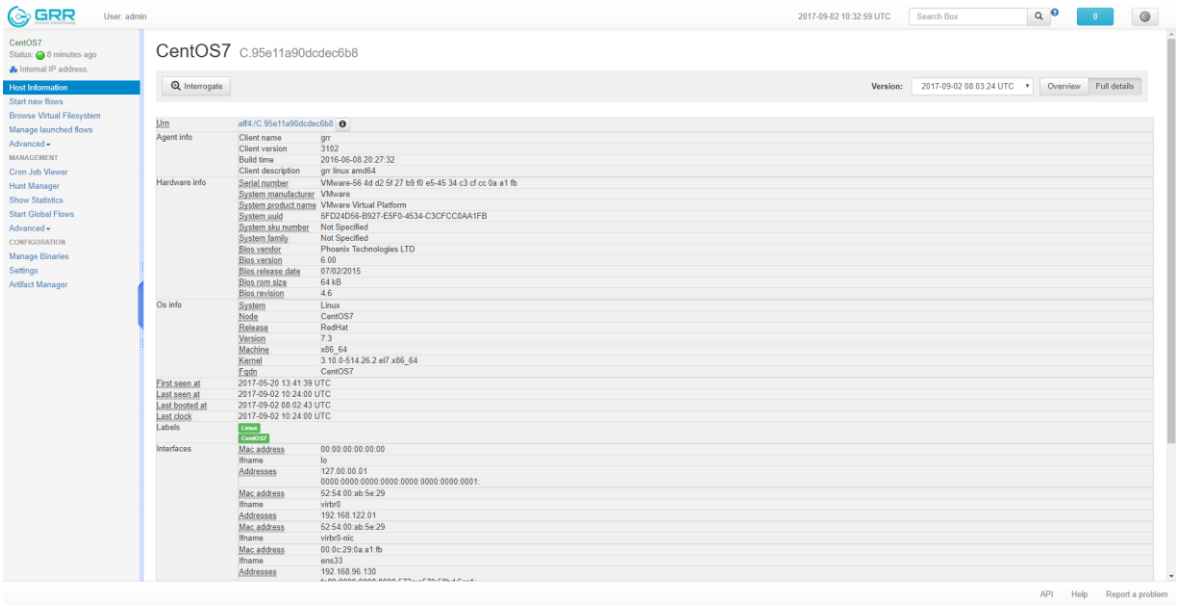
Slika 3 Pregled klijenata

Kako bi se dobio prikaz klijenata kao u Slika 3 Pregled klijenata, potrebno je napraviti pretragu u polju označenom crvenim okvirom bez filtera, odnosno teksta pretrage. Takva pretraga otvara prikazanu listu svih klijenata sa njihovim osnovnim informacijama. Jedina informacija na ovoj slici koja je podešena od strane servera, odnosno korisnika na serveru, su oznake (*Labels*). One se postavljaju i uklanjaju za označene klijente putem gumba u

gornjem lijevom kutu konzole. Ostale informacije o stroju klijent dohvaća i šalje sa jednim od automatskih, dobro znanih tokova.



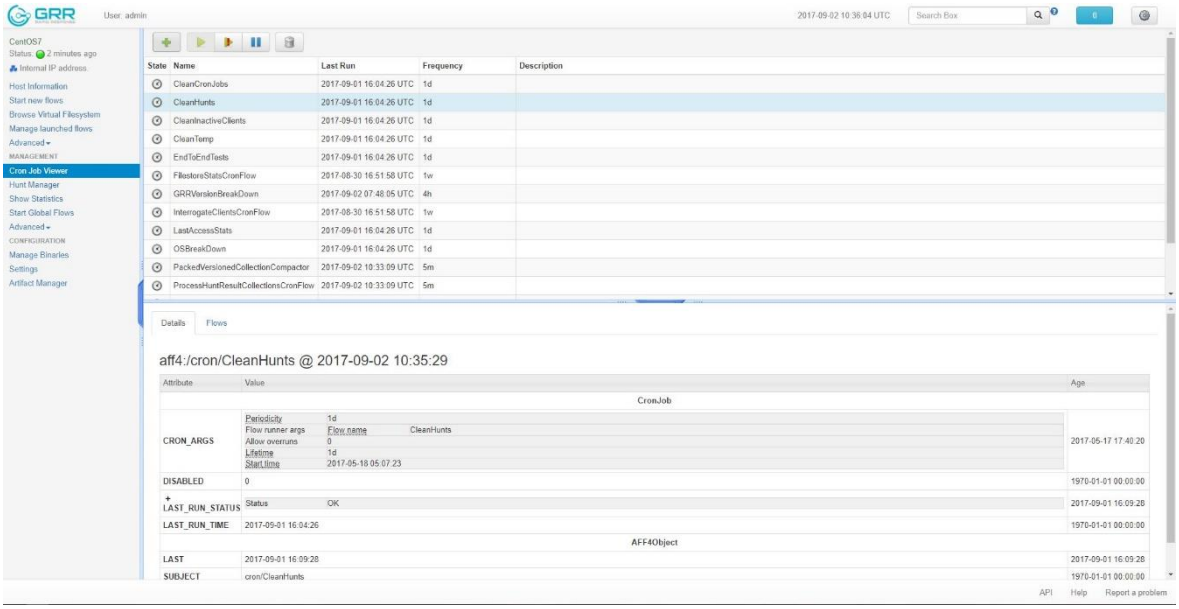
Slika 4 Detalji o klijentskom stroju



Slika 5 Puni detalji o klijentu

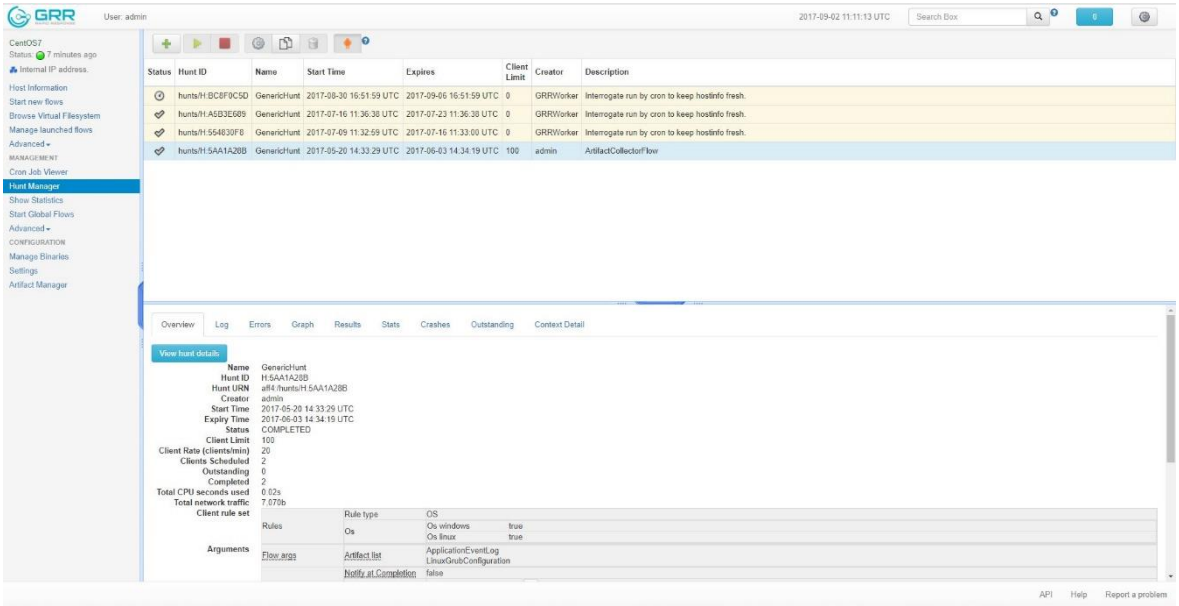
Na slikama Slika 4 Detalji o klijentskom stroju i Slika 5 Puni detalji o klijentu vidimo konzolu kako prikazuje detaljnije informacije o odabranom klijentu, dok u bočnoj traci (lijevo) dobivamo kontekstne informacije i akcije koje možemo izvršiti na klijentu. Opcije uključuju pokretanje novih tokova, pregled datotečnog sustava klijentskog stroja korištenjem FUSE sloja, upravljanje pokrenutim tokovima te napredne opcije koje se sastoje od pregleda zapisa GRR klijentskog programa. Ove opcije su korisne za pregledati ako se

klijentski proces ruši u radu ili ako želimo pregledati koliko GRR klijentski procesi opterećuju sustav. Vrsta operacijskog sustava ne utječe na dostupne opcije u bočnoj traci.



Slika 6 Pregled cron poslova

Slijedeći dio bočne trake su serverski zadaci grupirani u dio zvan upravljanje (*Management*). Prvi u nizu je preglednik cron poslova prikazan na Slika 6 Pregled cron poslova, odnosno zadataka koji se izvršavaju periodički na serveru. Gornji dio konzole sadrži listu svih poslova koji su definirani na sustavu dok se u donjem dijelu prikazuju detalji o odabranom poslu.



Slika 7 Upravitelj lovova

Upravitelj lovova (*Hunt Manager*), prikazan na slici Slika 7 Upravitelj lovova, sadrži prikaz nedavno izvršenih lovova, klikom ne jedan se prikazuju detalji u donjem dijelu. Zanimljivi

detalji koji se mogu vidjeti u detaljima lova uključuju grafikone izvršavanja, zauzeća računalnih resursa na klijentskim strojevima, pregled strojeva na kojima je proces lova najduže trajao ili zauzeo najviše resursa, kao i naravno rezultate lova. Rezultati zavise o prirodi lova, u slučaju sa slike to je bilo prikupljanje informacija o specifičnim datotekama na sustavu, odnosno artefakata. U tom slučaju rezultat lova će sadržavati podatke o datoteci, kao što su njezin kriptografski sažetak, vremena kreiranja, modifikacije i zadnjeg pristupa.

Slijedeća opcija prikazuje pregled statistika sustava, kao što su raspodjela klijentskih inačica GRR-a, pregled zadnjih aktivnosti klijentskih programa, raspodjelu operacijskih sustava u okruženju, te dodatne statistike vezane za sam server. Međutim, u vlastitoj instalaciji GRR-a nijedna od tih statistika nije radila ispravno. Pokušaj ispravljanja problema je prošao neuspješno, uključivao je pregled poruka o greškama vezanih uz odgovarajući modul te pregled svih grešaka u vrijeme otvaranja statistika. Pretraga na internetu u vrijeme pisanja nije vratila nikakve referente rezultate te, kako to ionako nije usko vezano uz fokus ovog rada, daljnji pokušaji otklanjanja ove greške su obustavljeni. Dok statistika ne predstavlja ključan dio GRR servera, poželjno ju je imati u većim implementacijama za brzi pregled stanja GRR okoline.

4.4. Konzola

GRR koristi interaktivnu Python (*ipython*) konzolu za izvođenje komandi. Konzola primarno služi za izvođenje akcija koje nisu definirane u web sučelju, kao što je na primjer definiranje novog lova ili toka. Naime, sve su akcije definirane u Python programskom jeziku, te ih je shodno tome prvo potrebno napisati pa spremiti tako da budu iskoristive iz web sučelja za ponovnu uporabu. Konzola ne sadrži nikakvu internu pomoć specifičnu za GRR, pa se mora osloniti na vanjske izvore znanja, što može predstavljati problem ako se GRR server postavi u izoliranu okolinu, gdje nema pristupa internetu.

Jedna od mogućnosti konzole je što se može podesiti da izvršavanje akcija na klijentu treba odobriti korisnički račun sa odgovarajućim privilegijama. Kako je konzola najmoćniji alat u radu sa GRR klijentima, ova mogućnost je više nego dobrodošla. Korisnik koji će koristiti konzolu mora kreirati token koji će potom koristiti sa svakom naredbom koju šalje klijentima. Token sadrži korisničko ime korisnika koji ga je kreirao i opis, odnosno proizvoljan tekst koji je namijenjen za kratki opis radnji zbog kojih se traži rad s klijentima. Token se odobrava samo jednom, od strane korisnika sa odgovarajućim privilegijama i

isključivo kroz konzolu. Jednako tako token se može opozvati, što je preporučljivo nakon što su obavljene tražene radnje. (Castle, et al., GRR User manual, 2017)

4.5. Predefinirani tokovi u sustavu

Ovo poglavlje će dati pregled zadanih tokova u GRR sustavu sa kratkim opisima:

- Interrogate – Pretražuje i vraća općenite podatke o stroju, koji se primarno koriste za popunjavanje detalja o stroju u pregledu detalja klijenata
- KeepAlive – Šalje zahtjev da klijent ostane aktivan određeni period vremena, odnosno u modu rada gdje aktivno očekuje naredbe sa servera
- OnlineNotification – Šalje email u trenutku kada se klijent pojavi u sustavu
- CacheGrep – Provjerava profilne direktorije web preglednika sadrže li datoteku sa traženim regularnim izrazom. Ova provjera može pregledati sadržaj Internet Explorer, Mozilla Firefox i Google Chrome web preglednika. Limit je maksimalno 50 rezultata po direktoriju.
- ChromeHistory – Prikuplja i analizira povijesni pregled (*history*) Google Chrome web preglednika.
- FirefoxHistory – Prikuplja i analizira povijesni pregled (*history*) Mozilla Firefox web preglednika.
- CheckRunner – Jedan od kompleksnijih tokova, radi slijedeće:
    - Identificira koje su provjere zadane za izvođenje na sustavu
    - Identificira artefakte koji se moraju dohvatiti da bi se provele te provjere
    - Organizira prikupljanje podataka o stroju
    - Predaje relevantne podatke provjerama
    - Vraća rezultate provjera u formatu za izvješćivanje
- AritfactCollectorFlow – Ovo je tok koji se bavi prikupljanjem artefakata sa klijentskih sustava. Ovaj tok ima dvije komponente, *ArtifactSources* i *ArtifactProcessors*. *ArtifactSources* definira moguće lokacije i tipove artefakata, na primjer datoteku ili set datoteka, vrijednosti u Windows Registry-u ili izlaze određenih naredbi u sustavu. Nakon prikupljanja, tok poziva *ArtifactProcessor* koji je namijenjen obradi dohvaćenih artefakata. Nakon obrade, podaci se mogu spremiti u AFF4 sustav ili predati drugom toku za daljnju obradu.
- DumpFlashImage – Služi za prikupljanje cjelokupne BIOS memorije.

- FileFinder – Na temelju zadanih parametara traži i dostavlja datoteke. Zamijenio je ranije tokove *FetchFiles, FingerprintFile* i *SearchFileContent*. Moguće akcije koje se mogu napraviti sa datotekama koje odgovaraju parametrima su stat (vraća podatke o datotekama), hash (računa i vraća kriptografski sažetak datoteka) te download (preuzima i šalje cjelokupne datoteke serveru).
- GetMBR – Služi za prikupljanje i slanje MBR-a (*Master Boot Record*).
- ListVolumeShadowCopies – Vraća popis *Volume Shadow Copy-a* (VSS). VSS je Microsoftov sustav za verzioniranje i čuvanje prijašnjih verzija datoteka ili volumena.
- AnalyzeClientMemory – Analizira klijentsku memoriju koristeći Rekall alate. Potrebno je toku definirati listu Rekall dodataka koji će se koristiti prilikom analize.
- MemoryCollector – Služi za skeniranje i izradu preslike memorije.
- Netstat – Vraća listu procesa koji imaju aktivne mrežne portove na sustavu koristeći Netstat
- ListProcesses – Vraća listu procesa koji se izvršavaju na sustavu.
- CollectRunKeyBinaries – Služi za prikupljanje programa koji su podešeni da se pokreću sa paljenjem sustava. Tok koristi *RunKeys* artefakt kako bi prikupio *RunKey* naredbe koje se koriste prilikom pokretanja programa. Tok zatim pogađa putanje do tih izvršnih datoteka i pokušava ih dohvatiti.
- GetMRU – Slaže listu nedavno korištenih datoteka za sve korisnike na sustavu i šalje je serveru.
- RegistryFinder – Traži objekte u Windows Registry-u koji odgovaraju zadanim kriterijima.
- RunReport – Izvršava traženi izvještaj i šalje rezultate na specificiranu email adresu.
- MACTimes – vraća vremena pristupa, modifikacije i promjene za datoteke koje se nalaze u virtualnom datotečnom sustavu.

5. Izvođenje tokova

Pokretanje definiranih tokova je jednostavno za izvesti koristeći web konzolu. Svi tokovi se nalaze u bočnoj traki kada je u glavnom prozoru sučelja odabrana klijentska instanca. Prilikom kreiranja toka bitno je zadati sve nužne parametre koji su potrebni da bi se određeni tok izvršio ispravno. Slijedi primjer izvođenja toka za izvlačenje povijesti Firefox web preglednika sa CentOS stroja.

Odabire se CentOS klijent i iz bočne trake *Start new flows* (Pokreni nove tokove) opcija. Iz kategorije web preglednika odabire se FirefoxHistory tok te se potom u glavnom dijelu sučelja prikazuju parametri toka. Neki su dijeljeni dok su drugi specifični za ovaj tok.

Pathtype određuje kako će se pisati putanja do datoteka, odnosno hoće li se koristiti Windows ili Linux, odnosno Unix putanje. Zadana opcija OS će očitati vrijednost iz klijentskog izvještaja dobivenog dobro znanim tokom te će koristiti adekvatne putanje. *Get archive* opcija određuje hoće li se skupljati arhivni podaci, odnosno podaci stariji od tri mjeseca. *Username* se odnosi na korisnika na klijentskom operacijskom sustavu i koristi se da ograniči tok na samo određene korisnike na sustavu. Pomoću ove vrijednosti GRR slaže putanju do datoteke, jer se ovi podaci nalaze u korisničkim direktorijima na operacijskom sustavu. *History path* opcija omogućuje eksplicitno navođenje putanje do direktorija sa datotekama koje sadrže podatke o stranicama koje je korisnik posjetio koristeći web preglednik. Poželjno ju je navesti i kad nije mijenjana, a predstavlja obavezan parametar ako je došlo do promjene smještaja tih datoteka.

Notify at completion opcija prikaže obavijest u samom web pregledniku (radi isključivo sa Google Chrome web preglednikom) ili slanjem emaila ako je tako podešeno u općim postavkama kada tok uspješno završi. Zadana opcija je da se ne šalju nikakve obavijesti o uspješno izvršenim zadacima. Neuspješno završeni zadaci uvijek trebaju prikazati obavijest i poslati email poruku.

Napredne opcije uključuju razinu prioriteta sa kojom bi se tok trebao izvršavati, ograničenje u sekundama CPU vremena koliko se tok smije izvoditi, ograničenje na ukupnu količinu korištenog mrežnog prometa između klijenta i servera, vrijeme kada bi tok trebao započeti te smještaj rezultata u AFF4 shemu.

Write intermediate results opcija će u rezultate toka zapisati svaku poruku koju klijent pošalje prilikom izvršavanja. Većina tokova se sastoji od nekoliko koraka gdje se kraj

svakog koraka komunicira sa serverom, no ti međukoraci ne sadrže konačne informacije koje tok mora naći. Sa ovom opcijom se u konačne rezultate može zabilježiti cjelokupna komunikacija klijenta i servera, što može biti korisno ako je potrebno otkriti uzrok problema sa određenim tokom.

Require fastpoll opcija šalje informaciju klijentu treba li nakon primanja inicijalne poruke ući u *fastpoll* način rada, gdje aktivno sluša i očekuje dodatne naredbe od strane servera. Ovu opciju se ne preporuča mijenjati jer može imati negativan utjecaj na performanse izvođenja toka.

Zadnja kategorija se tiče izvoza rezultata. GRR dolazi sa nekoliko mogućih načina izvoza podataka tokova. Svaki rezultat se zapisuje u AFF4 shemu, međutim, često zna biti korisno rezultate izvući iz sustava, bilo u svrhe kreiranja internih izvještaja, slanje rezultata drugim analitičarima koji možda nemaju pravo pristupa na sustav ili u za vanjsku arhivu. Podržani formati izvoza su CSV (*Comma Separated Values,* Zarezom odvojene vrijednosti) datoteka, izvoz u email kompatibilni format sa automatskim slanjem tog emaila na adrese definirane u sustavu, te u format kompatibilan sa Google BigQuery analitičkim skladištem, jednim od Googleovih big data rješenja.

Pored odabira formata izvoza, izvoz se može dodatno prilagoditi tako da se umjesto slanja samo izvještaja o provedenom toku mogu poslati i dohvaćene datoteke. Ova opcija dodatno opterećuje sustav jer zahtijeva dodatnu obradu podataka, pošto se dodaci za izvoz pokreću tek nakon što se tok završio, ova opcija zahtijeva da se dohvaćenim datotekama ponovno pristupi kako bi se mogle poslati zajedno sa rezultatima. Naravno, odabir ove opcije može znatno utjecati na veličinu izvozne datoteke, zavisno o tome što je sve tok dohvatio.

Follow URNs opcija će pored AFF4 adresa rezultata pokušati dohvatiti i objekt na kojeg adresa pokazuje. Kao i prethodna opcija, ovo zahtijeva ponovni pristup AFF4 objektima kako bi se isti mogli dohvatiti i ima negativni utjecaj na performanse toka.

Opcija koja izvozu podataka dodaje kriptografske sažetke će dodati sažetak samo ako je isti bio izračunat od strane samog toka, dakle sažetak se ne računa prilikom izvoza što znači da pojedini tokovi neće imati ove podatke u izvozu.

Zadnja opcija koja se može dodati su anotacije, čija je primarna namjena da se napravi razlika između višestrukih izvezenih podataka. Na primjer, definiramo jednu anotaciju za tokove i lovove koji se izvode automatski, a drugu za tokove i lovove koji su pokrenuti ručno.

Jednom kad su sve nužne opcije definirane, klikom na *launch* tok se pokreće, odnosno dodaje na listu za izvođenje ako smo odabrali vrijeme u budućnosti kao vrijeme početka toka. Tok kreće na izvođenje i na samom serveru nema povratnih informacija o stanju toka na klijentu sve dok isti ne završi uspješno ili neuspješno. Ako je uključena opcija da se prikazuju međurezultati, oni će biti upisani kao jedna od vrijednosti unutar samog lova. Ovisno o kompleksnosti toka i drugim parametrima, kao što su na primjer višestruki zakazani tokovi koji se izvode na klijentu, na rezultate se može čekati duže vrijeme.

Web konzola ima grešku gdje često ne osvježi status tokova na glavnom pregledu, već su točni podaci o statusu toka samo na pregledu detalja samog toka. Čak i kad se odabere određeni tok koji na glavnom pregledu ima oznaku da je još uvijek u tijeku, u detaljima je uredno prikazan kao gotov. Na žalost, u višestrukim navratima je primijećeno da nakon pregleda detalja, glavna lista tokova još uvijek prikazuje krivi status, što treba imati na umu prilikom korištenja web sučelja, Uzrok ove greške autoru nije poznat, niti je našao ikakav prijedlog rješenja.

6. Izvođenje lova

Izvođenje lova nije znatno drugačije od izvođenja toka, jer lov nije ništa drugo osim tokova koji se paralelno izvode na više strojeva u okolini. To znači da su početni koraci konfiguracije lova i konfiguracije toka identični. Proces počinje sa odabirom i konfiguracijom željenog toka, te konfiguracijom načina izvoza rezultata. Kod kreiranja lova se potom javlja dodatan korak, a to je kreiranje pravila koja će se pratiti za odabir klijenata na kojima će se lov izvršavati. Zadana postavka je da nema pravila, što znači da će se lov izvršiti na svim klijentima u okolini.

Moguće je definirati nekoliko pravila koja se mogu povezati logičkim i ili logičkim ili operatorima (*match all* i *match-any* u sučelju). Pravila se može dodati proizvoljan broj što omogućuje veliku razinu granularnosti. Zadana opcija po kojoj se određuju klijenti je filtriranje po operacijskom sustavu, gdje se može odabrati hoće li se lov izvršiti na operacijskom sustavu Windows, Linux ili OS darwin, odnosno OS X. Može se odabrati jedan, dva ili sva tri operacijska sustava, premda ova zadnja opcija anulira filtriranje.

Druga opcija uključuje korištenje oznaka koje su definirane od strane korisnika u GRR web sučelju. Kako su te oznake proizvoljne, one su često najkorisnije za filtriranje lova, pod uvjetom da su svi klijenti u sustavu uredno označeni. Oznaka za filtriranje se može dodati proizvoljna količina te ih primijeniti na klijente koji odgovaraju oznakama, gdje su višestruke oznake u logičkoj i ili logičkoj ili vezi, ili na klijente koji ne odgovaraju oznakama, gdje su te oznake povezane logičkim i ili logičkim ili operatorima.

Slijedeća opcija je korištenje regularnih izraza i tu se podrazumijeva korištenje AFF4 strukture za odabir klijenata. Kako je ranije napomenuto, svaki objekt u AFF4 strukturi dobiva svoju globalno jedinstvenu oznaku. Sa regularnim izrazom može se odrediti na kojim će se klijentima lov pokrenuti, koristeći bilo koji postojeći AFF4 objekt, odnosno URN u AFF4 strukturi. Ovaj način odabira klijenata može biti praktičan kada se ne zna točno koji klijenti posjeduju neki od objekata potrebnih za lov, odnosno istragu, jer se lov može primijeniti samo na klijente koji sadrže taj AFF4 objekt, bez znanja koji su to točno klijenti.

Zadnje pravilo po kojem se može vršiti filtriranje je korištenjem vrijednosti (u sučelju *Integer*). Ovo pravilo se također oslanja na AFF4 strukturu, no ovaj put se ne pretražuju AFF4 objekti, već vrijednost koja se pohrani u AFF4 objekt, odnosno URN AFF4 strukture. Korištenje ovog filtra zahtijeva izuzetno poznavanje AFF4 strukture na kojoj GRR počiva.

Korištenjem ovog filtra može se postići najviša stopa granularnosti odabira klijenata, međutim ona je također najteža za implementirati, upravo zbog potrebe za iscrpnim znanjem cjelokupne AFF4 strukture kao i vremenom koje je potrebno da se tražene vrijednosti inicijalno utvrde za primjenu u pravilo.

Vlastite opaske i prijedlozi prilikom korištenja sustava su da je preporučljivo koristiti filtriranje po operacijskom sustavu ili vlastitim, općenitijim oznakama za općenite istrage, kada je potrebno prikupiti snimku stanja ili pretražiti sve tražene artefakte specifične za dani operacijski sustav. Urednim i preciznim vođenjem oznaka se također može kvalitetno ograničiti obujam lova koji se izvode u okolini. Oznake su proizvoljne te mogu biti općenite kao na primjer „Windows Server 2012“ do preciznih kao na primjer „IIS server“. Korištenjem oznaka se najlakše može ograničiti lov na tražene strojeve ili na specifične artefakte koji su rezultat pojedine aplikacije koja se nalazi samo na određenim strojevima kojima je dodijeljena adekvatna oznaka.

Opcije filtriranja po regularnim izrazima i po vrijednostima AFF4 objekata bi prepustio isključivo naprednim korisnicima u situacijama kada je apsolutno nužno ograničiti lov na izrazito precizan set klijenata, bilo zbog osjetljivosti ili opterećenja koje će taj lov izazvati. Treba imati na umu da prilikom izrade lova nema koraka gdje će se prikazati lista klijenata na kojima će se lov izvršiti, što ove zadnje dvije opcije čini dosta problematičnima jer nema provjere rezultata, a greška u unosu može rezultirati situacijom gdje se lov neće pokrenuti na nijednom sustavu, potencijalno na svim sustavima ili na krivom setu sustava. Po vlastitom mišljenju, bilo bi moguće ovu fazu dodati u kreiranje lova, međutim, kako ona počiva na drugom toku, odnosno u ovom slučaju lovu (*checkRunner* konkretno, no on može također pozvati dodatne tokove), moglo bi se čekati na rezultate puno više nego što je praktično.

Stoga bi sa tim opcijama predložio dodatan oprez, te uređivanje oznaka pomoću kojih većina specifičnih uvjeta može biti implementirana bez potrebe za provjerom AFF4 strukture. Mogu biti korisne kad se mora provjeriti za određenu datoteku ili drugi trag koji ostavlja određeni maliciozni program, ali i tom slučaju je poželjno provjeriti sva odgovarajuća računala, bilo po mrežnom segmentu ili po operacijskom sustavu za postojanje i status traženog artefakta.

6.1. Izvođenje lova po vremenskom rasporedu

Lovovi se mogu namjestiti da se pokreću periodički. Na ovaj način možemo dobiti informaciju o novim događanjima na sustavu, vidjeti i analizirati nove procese koji su se izvršavali između dva lova, ili vidjeti jesu li neki od artefakata ili nadziranih datoteka izmijenjeni. Ovo je jedna od korisnijih opcija jer su takve akcije prikupljanja liste procesa ili osjetljivih datoteka, pod uvjetom da same datoteke nisu velike, ne rezultira prevelikim negativnim utjecajem na sustav, dok se postiže povijesno praćenje što može znatno pomoći jednom kad se utvrdi da je došlo do neželjenih akcija. Međutim, treba imati na umu da ovo nije zamjena za bilo kakvu zaštitu od malicioznih radnji ili programa. GRR je isključivo alat za nadziranje i utvrđivanje uzroka i opsega štete nakon što se dogodila.

Prilikom postavljanja periodičkog lova, traži se opis lova koji će stajati u pregledu svih zadataka koje server izvodi periodički. Postavlja se period koliko često će se lov pokretati kao i koliko može biti aktivan prije nego se prekine. Na taj način se lov koji je namješten da se izvodi jednom tjedno može prekinuti ako traje duže od nekoliko sati. Zadnja specifična opcija koja se može postaviti je *Allow overruns*. Ako se opcija uključi, prijašnji lov neće biti prekinut prije pokretanja novog. Ova opcija je poželjna ako je recimo period izvođenja jako kratak te se može dogoditi da će novi lov biti pokrenut dok stari nema mogućnost završiti, a rezultati su bitni. Nakon ovog dijaloga, lov se dalje postavlja jednako kao i običan.

Bitna napomena kod ovako zakazanih lovova je da se ne prikazuju automatski u upravitelju lovova (*Hunt manager*). Mora se odabrati opcija Prikaži/sakrij automatizirane lovove (*Show/hide automated hunts*). Ova opcija će prikazati korisničke lovove koji su namješteni na periodičko pokretanje, ali i lovove koje automatski pokreće GRR server u sklopu svog normalnog rada. Za razaznavanje korisničkih od sistemskih lovova preporuča se jedinstvena shema opisa lovova koja se razlikuje od sistemske.

Vremenski zakazani lovovi su koristan alat, ali jedan koji je potrebno koristiti samo koliko je nužno te isključivo za kritične sustave i objekte. Naime, zakazani lovovi se pokreću u zadano vrijeme, ne postoji mehanizam odgode u slučaju da je klijentski sustav opterećen drugim procesima, koji su u danom trenutku kritičniji za rad. Idealna opcija je zakazivanje lovova u vremenima kada se očekuje manja aktivnost sustava, međutim to također povlači sa sobom da je period u kojem se može dogoditi izmjena nadziranih objekata znatno veći i dok je tragove teško (ali ne i nemoguće) sakriti. Potencijalno može proći puno vremena od promjene do detekcije.

Prednost je što će se lov izvršiti neovisno o tome koliko je klijent bio neaktivan ili mrežno nedostupan, jer se svi lovovi čuvaju suspendirani na serveru sve dok se klijent ne javi. Premda to znači da klijent sam neće izvoditi lovove u periodima kada je prekinuta komunikacija sa serverom, neće biti potrebne nikakve dodatne radnje jednom kad se komunikacija ponovno uspostavi.

7. Greške

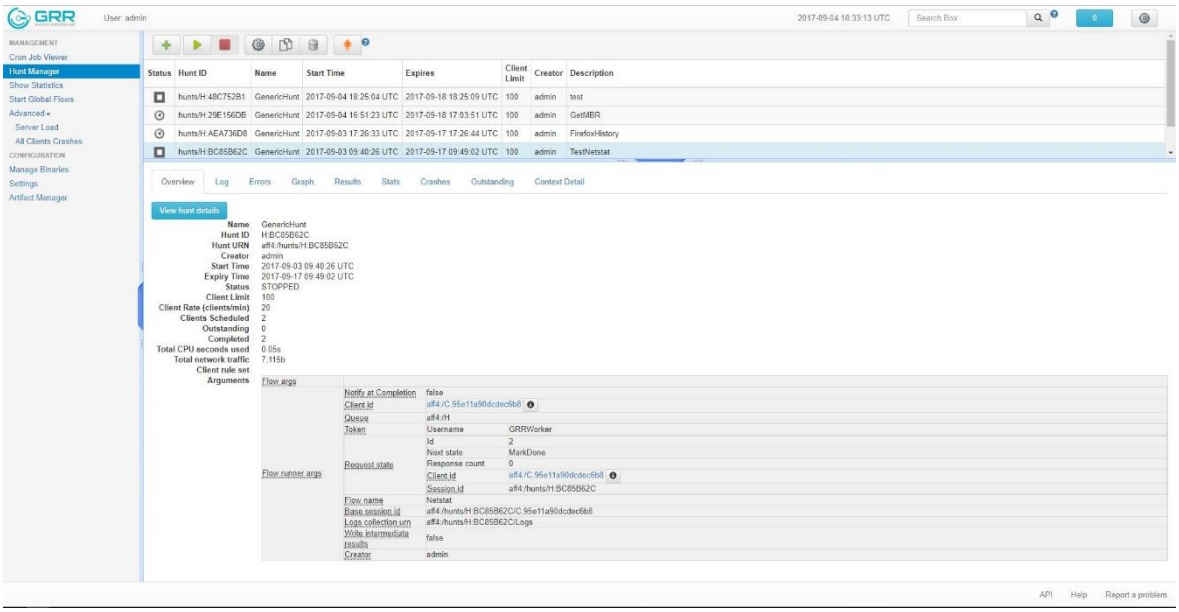
7.1. Lovovi/tokovi

Nažalost, primijećene su brojne greške vezane uz lovove koji se pojavljuju u web sučelju servera. Lovovi su najkompleksnija zadaća koju GRR izvršava te je i logično da se upravo kod njih događa najveći broj grešaka.

Već je ranije spomenuto da lovovi na glavnom pregledu upravitelja lovova ne prikazuju ispravan status. Statusi u kojima lov može biti su prikazani ikonama, slijede njihovi opisi:

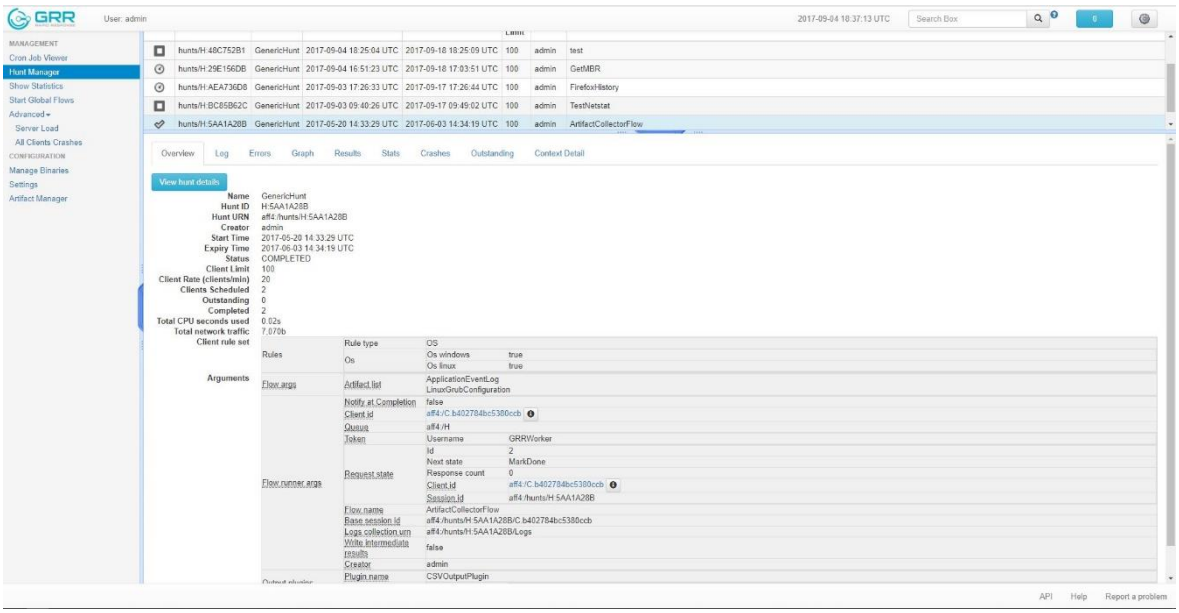
- Pauziran – Lov čeka na izvođenje. Jednom pokrenut, lov se može jedino prekinuti, ne i privremeno zaustaviti.
- U tijeku – Lov je pokrenut i čeka na rezultate, odnosno na povratnu poruku da je lov gotov.
- Zaustavljen – Lov koji je zaustavljen prije nego je sam završio izvođenje. Lov u tom stanju se može ponovno pokrenuti jer njegov unos ostaje u upravitelju lovova, ali će uvijek krenuti iz početka na svim klijentima, neovisno je li na nekima uspješno završen.
- Gotov – Lov je uspješno završio na svim klijentima.
- Greška – Lov je završio s greškom na svim klijentima. Teoretski bi trebalo biti moguće samo sa izrazito loše definiranim lovom ili postavljanjem prekratkog vremena izvođenja.

Najčešći problemi se javljaju sa statusom u tijeku, koji ostaje aktivan u upravitelju lova premda je sam lov prešao u drugo stanje.



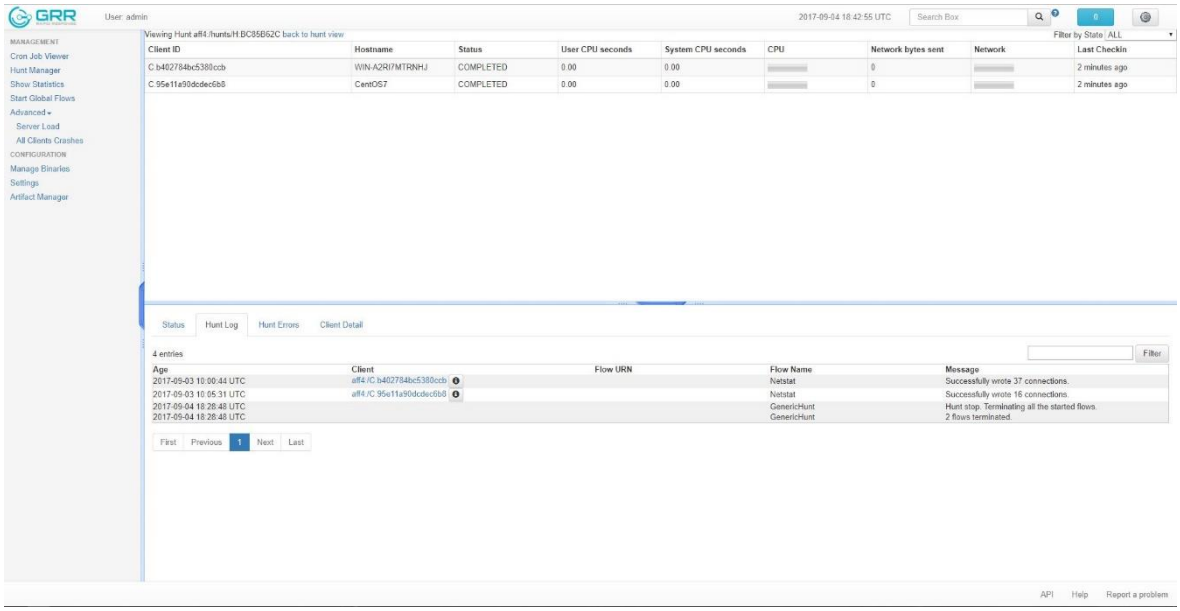
Slika 8 Pregled Netstat lova iz Upravitelja lovova

Za primjer je iskorišten jedan relativno jednostavan lov koji je samo popisao rezultate *netstat* naredbe na oba stroja u okolini. Lov ima status „Stopiran“ nakon što je ručno zaustavljen iz konzole uslijed čudnog ponašanja. Pogledom na detalje lova iz Slika 8 Pregled Netstat lova iz Upravitelja lovova vidimo u argumentima da je *RequestState* argument u identifikatoru 2 i da je slijedeće stanje *MarkDone*, odnosno „označi kao gotovo“. Premda bi bilo logično za zaključiti da je izvođenje lova naišlo na grešku prije nego je moglo uspješno zapisati konačni status, to je kriva pretpostavka.



Slika 9 Lov - Prikupljanje artefakata

Naime, pregledom detalja lova koji je uspješno završio na Slika 9 Lov - Prikupljanje artefakata vidimo da je i njegov *RequestState* sa identifikatorom 2 i da je slijedeći status *MarkDone*. Ovo je primjer lova koji je također pokrenut na oba stroja u okolini ali sa kompleksnijim zadatkom prikupljanja artefakata, specifično GRUB konfiguraciju (*GNU Grand Unified Bootloader,* zadužen za podizanje operacijskog sustava nakon što BIOS odradi provjere hardvera) te aplikacijski segment Windows event loga.

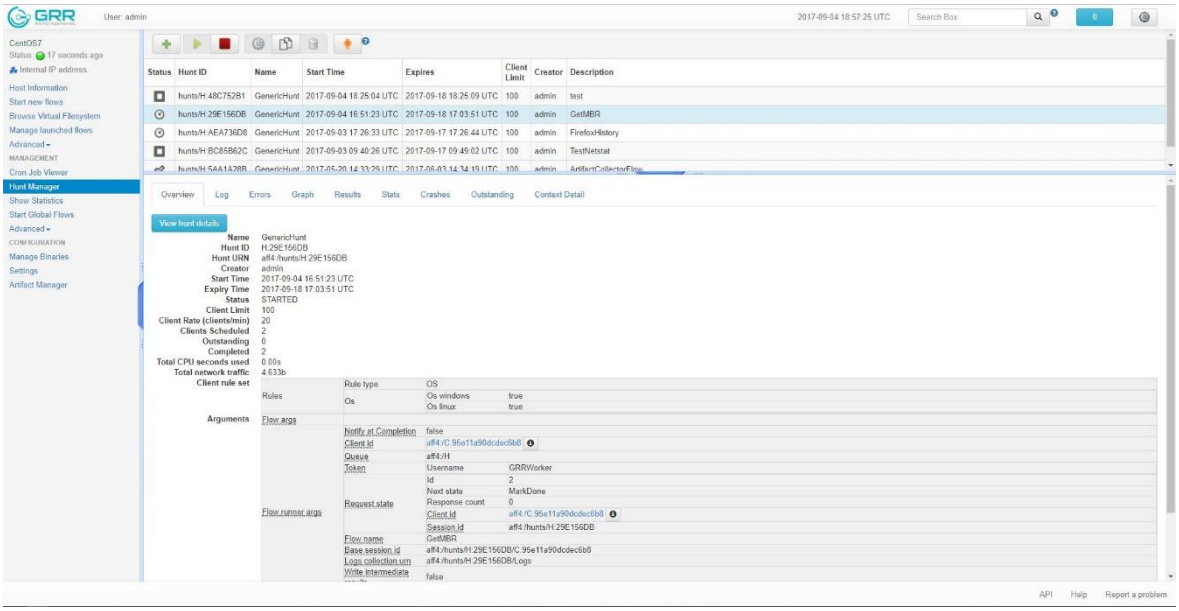


Slika 10 Nestat lov - Detalji

Pregledom detalja lova s greškom vidljivom na Slika 10 Nestat lov - Detalji, vidimo da se na oba stroja naredba uredno izvršila te da su rezultati poslani i zapisani u AFF4 strukturu. Greške lova (*Hunt Errors*) ne prikazuju nikakve greške sa samim lovom. Pregled zapisa sustava direktno ne pokazuje nikakve očite greške sa izvođenjem lova. Isti se čini kao da čeka na neko neodređeno slijedeće stanje i da je ostao u suspendiranom stanju aktivan na sustavu. Pregledom samog toka koji je definiran u Python programskom jeziku nije otkrivena nikakva greška, ali treba opet napomenuti da autor ima samo osnovno poznavanje tog programskog jezika. Međutim, izvođenje tog toka samo kao tok na jednom stroju završi uspješno. Točan uzrok problema stoga ostaje nepoznat, međutim kako se radi o toku koji je definiran od strane razvojnih inženjera aplikacije koji nema nikakvih dodatnih parametara koji bi mogli utjecati na rezultat, krajnje je čudno zašto taj lov ostane u statusu „U tijeku“.

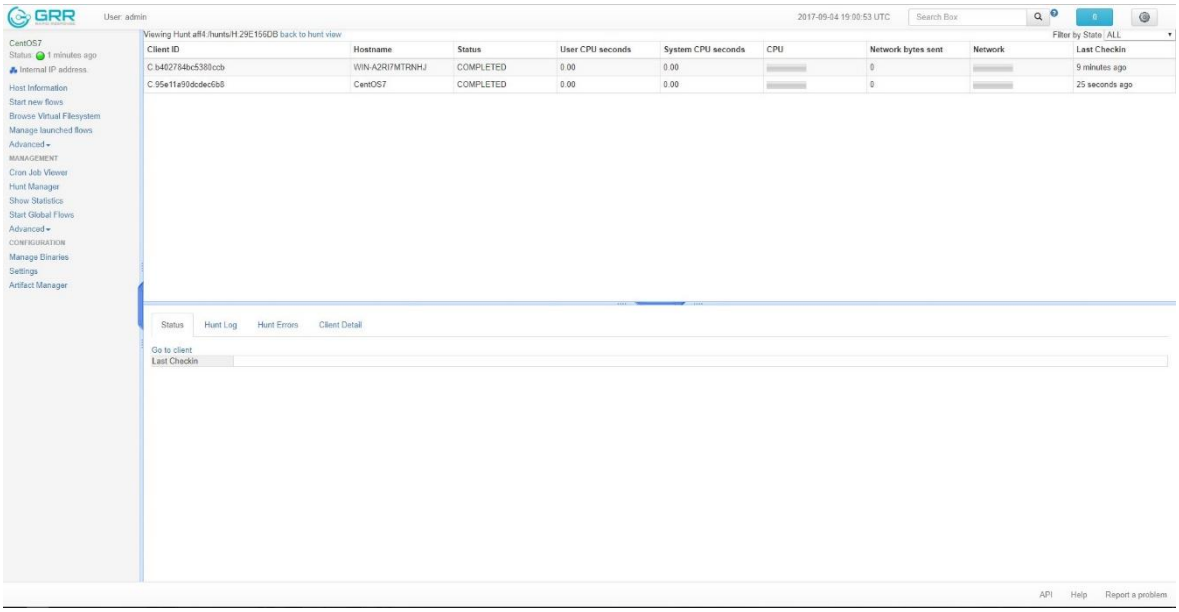
No, nažalost, to nije jedini tok koji ostaje u krivom statusu. Naime, uočeno je nekoliko lovova koji su također ostali u statusu „U tijeku“ premda su ustvari prekinuli sa greškom na klijentskim sustavima. Vrlo čest slučaj je kad se na jednom stroju lov uspješno završi dok je

na drugom došlo do greške prilikom izvođenja. Ovo je puno gori scenarij, jer će proći neko vrijeme prije nego što osobi koja nadzire konzolu postane jasno da je došlo do greške, posljedice koje se dodatno povećavaju ako je u pitanju lov za koji se očekuje da traje duže vremena.



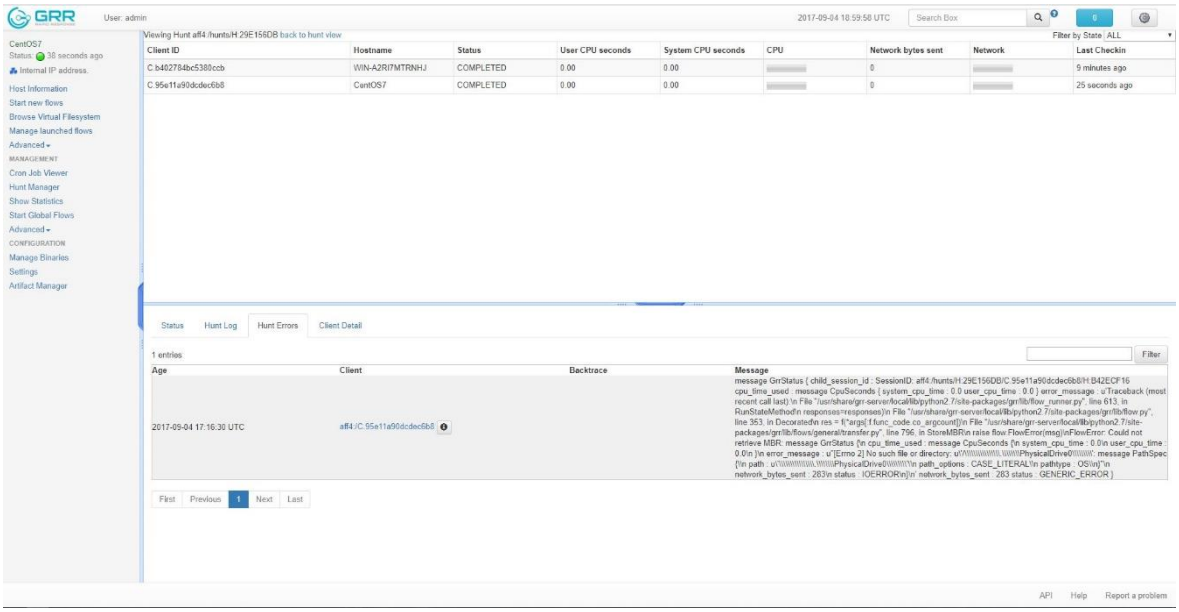
Slika 11 Lov - prikupljanje MBR-a

Primjer je lov sa Slika 11 Lov - prikupljanje MBR-a, u kojem je pokrenut lov koji dohvaća MBR (*Master Boot Record*, sektor koji sadrži podatke nužne za podizanje operacijskog sustava). Na slici vidimo slično stanje kao i kod ranijeg lova, gdje je naizgled lov završio samo se nije prebacio u gotovo stanje.



Slika 12 Lov - prikupljanje MBR-a, status

Čak i pregled detalja vidljiv na Slika 12 Lov - prikupljanje MBR-a, status ne prikazuje nikakve probleme sa lovom, oba klijenta javljaju status *COMPLETED* što bi pretpostavljalo da je na oba klijenta lov uspješno izveden.



Slika 13 Lov - prikupljanje MBR-a, detalji

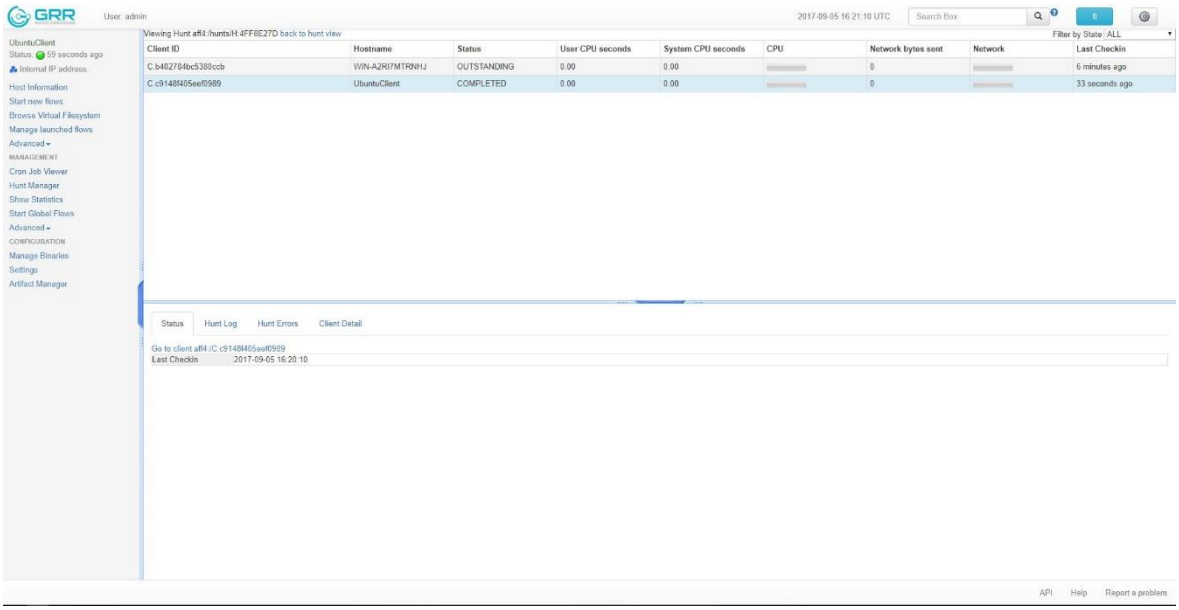
No, pregledom grešaka lova (*Hunt errors*) na Slika 13 Lov - prikupljanje MBR-a, detalji, vidimo da je ustvari lov CentOS klijentu (klik na gumb pored AFF4 urn-a pod *Client* daje pregled detalja o klijentu gdje se može vidjeti da se radi o CentOS stroju) završio sa greškom. Poruka o grešci ukazuje da MBR nikad nije prikupljen jer isti nije nađen. Dakle, tek na ovom ekranu je vidljivo da lov ustvari ima grešku zbog koje sa jednog od klijenata nije prikupljena datoteka čije je prikupljanje smisao cijelog lova. Ovakvo netransparentno prikazivanje grešaka koje se dogode prilikom izvršavanja lova može dovesti do velikih problema ako nije na vrijeme uočeno u produkcijskoj okolini kada se dogodi sigurnosni incident. Naime, u tom slučaju će lov za koji se mislilo da je uspješno završen biti utvrđeno da nije prikupio informacije u kritičnom trenutku.

Premda bi se moglo argumentirati da je bilo koji lov koji ne završi sa statusom *Completed* potrebno pregledati, kako su takvi lovovi činili većinski udio tijekom pisanja ovog rada, vrlo je vjerojatno da bi se u bilo kakvoj aktivnijoj produkcijskoj okolini takvi krivi statusi počeli previđati sve do trenutka kad bi došlo do incidenta. Učestalost ove greške, pogotovo prilikom izvođenja jednostavnih lovova sa malo ili bez dodatnih argumenata koji utječu na izvođenje lova je razlog za brigu. U slučaju prikupljanja MBR-a, operacijski sustav u pitanju je

CentOS, popularna besplatna alternativa RedHat Enterprise Linuxu koju je autor susreo na velikom broju produkcijskih servera u svojoj karijeri.

Sve ove greške su otklonjive zahvaljujući otvorenom kodu samog servera i netko sa naprednijim znanjem Python programskog jezika bi ih zasigurno mogao otkloniti. Međutim, zbog činjenice da se čak i osnovnije provjere ne uspijevaju izvršiti kako treba, te zbog izrazito loših metoda obavještavanja korisnika sustava o kritičnim radnjama, postaje upitno koliko se može osloniti na rad ovog alata. S druge strane, ako se odluči ručno provjeravati svako izvođenje svakog lova na sustavu, to znatno podiže opterećenje osobe koja je zadužena za upravljanje sustavom te se uopće nameće pitanje iskoristivosti kada se uzme u obzir postojanje konkurentskih alata, pa čak i komplementarnih alata koji imaju neke od traženih funkcionalnosti.

Naknadno je, nakon greški i drugih lovova na CentOS operacijskom sustavu, isti bio zamijenjen sa novom virtualnom mašinom koja je sadržavala operacijski sustav Ubuntu, međutim veliki broj grešaka koji se pojavljivao na CentOS-u je također javljao greške i na Ubuntu operacijskom sustavu. Uzrok ovih grešaka je nepoznat, pretraga na internetu nije dala nikakve konkretne odgovore zašto se isti događaju za relativno jednostavne tokove, odnosno lovove na relativno raširenim i poznatim Linux distribucijama. Zamjena je samo istaknula novu nekonzistenciju alata gdje je ponovljen lov koji dohvaća MBR sa Windows stroja i sa novo dodanog Ubuntu stroja.



Slika 14 Lov - Prikupljanje MBR-a, novi

Kao što je vidljivo na Slika 14 Lov - Prikupljanje MBR-a, novi, lov javlja da je na Ubuntu klijentu završio, dok je na Windows klijentu u statusu *OUTSTANDING* koji upućuje na problem sa izvođenjem. Međutim, pregledom detalja lova na oba klijenta, oba imaju identičnu (provjereno alatom za usporedbu teksta) poruku o grešci. Unatoč tome što je jedan od klijenata u statusu greške, lov se još uvijek prikazuje kao da je u tijeku na ekranu upravitelja lovova. Pokušaj otkrivanja uzroka ove nelogične ne konzistencije nije urodio plodom.

Čak i tokovi koji su vezani na vanjske programe, kao recimo tok za prikupljanje memorije, također završe s greškom, premda su testne virtualne mašine konfigurirane sa jako malo radne memorije (2 GB), međutim kod njih se jave poruke da je klijent terminiran. Nažalost, nisu puno opširnije od toga pa je teško zaključiti je li klijenta ubio proces dadilja, što ne bi bilo poželjno jer to bi značilo da proces dadilja ne uzima u obzir tok koji se izvršava, ili je proces terminirao operacijski sustav iz nekog nepoznatog razloga. Sistemski zapisi operacijskih sustava na kojima se tok izvršavao nemaju nikakve vezane poruke o grešci u vrijeme kada se tok izvodio.

Treba uzeti u obzir da je moguće da je uzrok grešaka korištenje virtualizacijske tehnologije. Dok je to sasvim moguć uzrok, istovremeno dovodi u pitanje iskoristivost rješenja gledajući kako trendovi danas sve višu naginju skoro cjelokupnoj virtualizaciji svih poslužitelja, ako ne već i radnih stanica. Ako se GRR ne može nositi sa virtualiziranim rješenjima, njegova primjena je posljedično veoma limitirana. Opet, nedostatak materijala na internetu ili u bilo kojem obliku vezanog uz GRR drastično ograničava mogućnost nalaženja konkretnih odgovora. Nema mnogo javno znanih korištenja GRR-a, dok Google sam priznaje da koristi znatno modificiranu verziju GRR-a interno, premda se tu primarno misli na pohranu AFF4 struktura i korištenje druge baze za pohranu, nije isključeno da modifikacija za kompatibilnost sa tim razlikama za sobom povlači i dodatne izmjene koje utječu na cjelokupan rad sustava.

7.2. Pokušaji otklanjanja grešaka

U radu sa GRR-om su se gore navedene greške redovito događale. Izuzev početnog podešavanja serverske komponente i instalacije klijenta, malo je koji aspekt korištenja alata prošao bez greški. Nažalost, kako dokumentacije za otklanjanje grešaka, kao i praktičnih iskustava sa korištenjem alata na internetu ima jako malo, postupak otklanjanja grešaka se

svodio na vlastite pokušaje utvrđivanja uzroka te implementacije raznih rješenja baziranih na vlastitom, limitiranom poznavanju programa i Python programskog jezika.

Cjelokupna infrastruktura je nekoliko puta podizana iz početka, počevši sa serverskom komponentom, kao i sa cijelim virtualnim mašinama. Isprobana je izmjena Python paketa na serveru prije instalacije GRR-a, unatoč tome što GRR server dolazi sa vlastitom verzijom Python okruženja koju koristi za rad. Ugašeni su svi vatrozidi između klijenata i servera, makar se nikad nije činilo da je komunikacija problem. Pregledavani su zapisi operacijskih sustava kao i zapisi GRR servera u nadi da će pokazati na neki konkretan uzrok problema, bilo općenito ili za određeni tok koji se neuspješno izvršio na klijentima. Poruke o greškama koje su tokovi izbacili su pretraživane na internetu sa vrlo slabo povezanim i nerelevantnim rezultatima. Dok se nekoliko puta i dogodilo da je pronađena slična greška na nekom od foruma ili web stranica za ispomoć, predložena rješenja ili nisu uopće bila primjenjiva na infrastrukturu ili ne bi ispravila grešku.

Jedan od glavnih problema je bila nekonzistentnost u radu. Često su uspješni tokovi prestali raditi, ili bi neki tokovi koji isprva nisu radili uspješno imali nekoliko uspješnih izvođenja. Varijable na sustavima su bile zadržane na minimumu, izuzev inicijalnog ažuriranja sustava prije instalacije klijenta, nikakva druga ažuriranja sustava nisu napravljena. Kasnije iteracije infrastrukture nisu uopće ažurirane nakon instalacije, međutim to nije promijenilo ishod rada.

Nekonzistentnost je dodatno otežavala postupak otklanjanja greška, jer se krenulo gledati potencijalne uzroke koji bi mogli biti aktualni samo u vrijeme izvođenja toka. Baš ta nekonzistentnost je dovela do prvog brisanja i reinstalacije cijele okoline, u nadi da je uzrok grešaka bio samo kumulativni rezultat višestrukih inicijalnih testova u periodu upoznavanja sa softverom. Međutim, kasnija izvođenja i greške su pokazale da to nije bio uzrok.

Kako je vrijeme prolazilo, mogućih metoda za ispravak greški je bilo sve manje. U toku rada je zamijenjen Linux sustav koji je davao više grešaka od Windows sustava korištenog u okolini. Premda CentOS operacijski sustav kao osnovu koristi RedHat Enterprise Linux, jedan od popularnijih sustava za poslovno korištenje, činjenica jest da je sa inačicom 7 i prelaskom na *systemd* dosta alata ostalo nekompatibilno, čak i 3 godine nakon izlaska. Međutim, ni Ubuntu operacijski sustav, čija je serverska inačica jedini podržani operacijski sustav za izvođenje GRR serverske komponente nije davala konzistentnije rezultate, niti manji broj grešaka.

8. Zaključak

Ovaj rad je inicijalno bio zamišljen kao istraživanje izvedivosti implementacije GRR-a u infrastrukturu jedne male ili srednje tvrtke iz regije, za implementaciju mogućnosti udaljene forenzike i provjera sustava u okolini, te za brži odgovor na incidente, koristeći ugrađene mogućnosti GRR-a. Jedna od važnijih stvari koja bi utjecala na ocjenu softvera je bila mogućnost jednostavne konfiguracije i održavanja, jer je početna pretpostavka bila da se prosječni zaposlenik u tvrtki te veličine ne može u cijelosti posvetiti održavanju i upravljanju GRR-om kao svojim jedinim zadatkom na poslu. Plan je potom bio procijeniti koliko je komplicirano dobiti „sliku stanja“ okruženja kroz GRR, koja bi služila kao početna točka, nakon koje bi se krenulo u simulaciju raznih zlonamjernih radnji s ciljem utvrđivanja koliko GRR može pomoći u detekciji i analizi pravog uzroka radnji.

Međutim, vrlo brzo nakon instalacije su uočeni problemi sa radom, odnosno izvođenjem aktivnosti koje čine srž funkcionalnosti GRR-a. Kako razni pokušaji uklanjanja tih greška nisu rješavali probleme, rad je sve više gubio taj inicijalni cilj, jer je tolika količina otklanjanja pogreški i nekonzistentnosti u radu na tako maloj i kontroliranoj okolini kao što je bila ova testna, automatski „diskvalificirala“ GRR kao alat koji bi se implementirao i koristio u zamišljenoj tvrtki.

Kako je rad bio istraživačke prirode, takav ishod je svakako bio moguć, te ga je možda i trebalo očekivati gledajući na nedostatak iskaza o korištenju alata na internetu. Većina članaka, kojih nema puno općenito, se primarno bavila opisima funkcionalnosti rada bez praktičnih primjera kojima bi pokazali funkcionalnost istih. Također, izrazito je malo članaka koji se bave greškama u radu, koje se od softvera u ovoj fazi razvoja skoro pa i očekuju. Sve to navodi na činjenicu da je ovo softver koji koristi jako mali broj korisnika koji su, gledajući općenitu opskurnost softvera, vjerojatno korisnici koji su sposobni prilagoditi osnovnu funkcionalnost softvera vlastitim potrebama.

GRR se nastavlja razvijati, no moguće promjene u budućnosti također ukazuju na nedostatak zrelosti softvera, jer se kao jedna od mogućih promjena spominje i prepisivanje cjelokupnog softvera sa Python programskog jezika na programski jezik Go. (Moser, Bushkov, Galehouse, & Lakomy, 2017)

Vlastiti dojmovi o GRR-u su da je to softver koji počiva na solidnim principima. U današnjoj okolini je sve više raznovrsnijih malicioznih napada na informatičku strukturu, gdje su

antivirusni alati sve nemoćniji da se bore protiv njih, a paralelno se sve više poslovanja koristi i informacija sprema na tu infrastrukturu. U tom okruženju, mogućnost brzog otkrivanja uzroka te vektora upada u nadziranu okolinu može drastično smanjiti vrijeme u kojem maliciozni program ili korisnik može počiniti štetu. Također, utvrđivanjem točnog uzroka tih radnji se može adekvatno zaštiti od istog ili sličnog napada u budućnosti, no to također zahtijeva i rješenje koje će biti stabilno, skalabilno i pouzdano u radu, što je nešto što se GRR nije pokazao u vlastitom testiranju. I dok je vrlo lako moguće da će se greške ispraviti, trenutno bi taj zadatak zahtijevao dugoročni angažman zaposlenika ili tima zaposlenika, što je nešto što si malo koja manja tvrtka može priuštiti, osim ako joj informatička sigurnost nije od apsolutne važnosti, a i onda postoji velika vjerojatnost da će se posegnuti za nekim komercijalnim rješenjem koje dolazi sa službenom podrškom proizvođača.

GRR je softver čije su mogućnosti obećavajuće i njegov razvoj bi trebalo pratiti. Međutim u trenutnom stanju nije spreman za uporabu bez puno izmjena i prilagodbi.

Popis kratica

GRR - *GRR Rapid Response*, također znan kao i *Google Rapid response*

GPO – *Group Policy Object*, Objekt Grupne Politike

urn - *Uniform Resource Name,* Ujednačeno ime resursa

TSK - *The Sleuth Kit*

NSRL - *National Software Registry List*, Nacionalni lista softverskog registra

FUSE – *File system in user space*, Datotečni sustav u korisničkom segmentu

MBR – *Master Boot Record*

VSS – *Volume Shadow Copy*

MRU – *Most Recently Used*, Nedavno korišteno

MAC – *Modify, Access, Create*, Izmjena, Pristup, Kreiranje

CPU – *Central Processing Unit*, Centralna Procesorska Jedinica

RAM – *Random Access Memory*, memorija nasumičnog pristupa

GRUB - *GNU Grand Unified Bootloader*

Popis slika

Slika 1 GRR konzola - Pregled klijenata 2

Slika 2 GRR Konzola 14

Slika 3 Pregled klijenata 14

Slika 4 Detalji o klijentskom stroju 15

Slika 5 Puni detalji o klijentu 15

Slika 6 Pregled cron poslova 16

Slika 7 Upravitelj lovova 16

Slika 8 Pregled Netstat lova iz Upravitelja lovova 28

Slika 9 Lov - Prikupljanje artefakata 28

Slika 10 Nestat lov - Detalji 29

Slika 11 Lov - prikupljanje MBR-a 30

Slika 12 Lov - prikupljanje MBR-a, status 30

Slika 13 Lov - prikupljanje MBR-a, detalji 31

Slika 14 Lov - Prikupljanje MBR-a, novi 32

Literatura

Carrier, B. (2017, 07 16). *About*. Retrieved from Sleuthkit: https://www.sleuthkit.org/about.php

Castle, G., Moser, A., Olson, B., mbushkov, Bilby, D., weslambert, . . . Wendt, D. (2017, 09 04). *GRR Administrator Documentation*. Retrieved from GitHub: https://github.com/google/grr-doc/blob/master/admin.adoc

Castle, G., Moser, A., Olson, B., mbushkov, Bilby, D., weslambert, . . . Wendt, D. (2017, September 04). *GRR User manual*. Retrieved from GitHub: https://github.com/google/grr-doc/blob/master/user_manual.adoc

Cohen, M., Garfinkel, S., & Schatz, B. (2009). Extending the advanced forensic format to accommodate multiple data sources, logical evidence, arbitrary information and forensic workflow. *Elsevier*, S58-S61.

Moser, A., Bushkov, M., Galehouse, B., & Lakomy, M. (2017). *GRR Meetup, Mar 2017*. Palo Alto: N/A.

Rekall Forensics . (2017, 07 14). *Rekall At a Glance*. Retrieved from Rekall Forensic: http://www.rekall-forensic.com/documentation-1/rekall-documentation/rekall-at-a-glance

Student vlastoručno potpisuje Završni rad iza zaključka s datumom i oznakom mjesta završetka rada te naznakom:

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada".

U Zagrebu, 7.9.2017..