BACHELOROPPGAVE:

PALMADNS

Et rammeverk for analyse av Passive DNS for deteksjon av ondsinnet nettverksaktivitet

FORFATTERE:

Andreas Moe

Lars Christian Andersen

DATO:

16.05.2014

SAMMENDRAG AV BACHELOROPPGAVEN

Tittel:	PalmaDNS	Nr: -
	Et rammeverk for analyse av Passive DNS for deteksjon av ondsinnet nettverksaktivitet	Dato: 16.05.2014
Deltakere:	Andreas Moe	
	Lars Christian Andersen	
Veiledere:	Erik Hjelmås	
Oppdragsgiver:	mnemonic AS	
Kontaktperson:	Andreas Bråthen	
Stikkord	Passiv DNS, Skadevare, Deteksjon, Fast-flux, DGA	
Antall sider: 177	Antall bilag: 7	Tilgjengelighet: Åpen

Kort beskrivelse av oppgaven:
Denne rapporten er sentralisert rundt Passiv DNS og analysen av dette til å kunne detektere ondsinnet nettverksaktivitet. Oppgaven leveres som et produkt til mnemonic AS.
mnemonic har idag en rekke sensorer for å overvåke nettverkstrafikken til bedrifter, og den oppsamlede informasjonen benyttes til deteksjon og videre etterforskning av reelle og potensielle sikkerhetshendelser.

Denne rapportens hovedleveranse er en innledende kartlegging og vurdering av potensielle attributer og egenskaper ved datasettet som finnes i passiv DNS, og hvordan dette kan potensielt benyttes til deteksjon av mistenklig og eller ondsinnet aktivitet. Ved denne kartleggingen har det blitt presentert en oversikt over 15 initielle analyserbare attributter og egenskaper ved passiv DNS-datasett. Videre er fem av disse gjennomgått og presentert på et detaljert nivå.

Denne teorien er videre brukt, i samarbeid med mnemonic, som grunnlaget til utvikle en kravspesifikasjon og designdokument for et rammeverk. Dette rammeverket skal kunne benyttes til å gjennomføre en analyse av de kartlagte attributene og egenskapene ved passiv DNS-data, og resultere i deteksjon av potensielt mistenkelig og eller ondsinnet nettverksaktivitet.

Etter utarbeidelsen av denne kravspesifikasjonen og designdokumentet er det utviklet en Proof-of-Concept av rammeverket via en implementasjon laget i programmeringsspråket Python. Ettersom dette kun er en Proof-of-Concept er det fremmet forbedringspotensialer og forslag til videre arbeid for å kunne ferdigstille en fullverdig applikasjon som kan settes i et produksjonsmiljø.

SUMMARY OF GRADUATE PROJECT

Title:	PalmaDNS	Nr: -
	A framework for analysis of Passive DNS for detection of malicious network activity	Date: 16.05.2014
Participants:	Andreas Moe	
	Lars Christian Andersen	
Supervisor:	Erik Hjelmås	
Employer:	mnemonic AS	
Contact person:	Andreas Bråthen	
Keywords:	Passive DNS, Malware, Detection, Fast-flux, DGA	
Pages: 177	Appendixes: 7	Availability: Open

Short description of the main project:

This report is focused around the area of Passive DNS and the analysis of this for detection of malicious network activity, and is a delivery to mnemonic AS. mnemonic has a collection of sensors monitoring the network traffic of several businesses, and uses the aggregated information for detection and further investigation of potential security incidents.

The main delivery of this report is a preliminary survey and assessment of potential attributes and properties of data sets collectable from passive DNS, and the usage of this information for detection of suspicious and or malicious activity. This survey has resulted in a list of 15 initial attributes and properties of a passive DNS data set, which all are analyzable. Five of these are further investigated and presented on a more detailed level.

This data is, in cooperation with mnemonic, used as a basis for the development of a requirement spesification and design document for a framework. The framework will be used for analysis of the presented attributes and properties of passive DNS, and should be able to detect suspicious and or malicious network activity.

As a further presentation of the requirement spesification and design document a Proof-of-Concept of the framework is implemented in the programming language Python. A Proof-of-Concept is but a basic example of the expected outcome, and such, this report presents elements for improvement and suggestions for future development needed to develop a functional application ready for production.

Forord

Det å skrive en bacheloroppgave er i våre øyne den perfekte avslutningen på en bachelorutdannelse. Vi ble gitt muligheten til å tre utenfor vår komfortsone og inn i nye utfordringer. Alt vi hadde lært de siste to og et halvt årene skulle kombineres for å produsere en så bra oppgave som mulig. Denne oppgaven har ikke bare trukket frem tidligere læring, men også tilført mange nye kunnskaper og gitt en dybdeinnsikt i det spennende temaet Passive Domain Name System (pDNS).

For at arbeidet med denne oppgaven gikk så bra, og var så lærerikt som det endte opp med, vil vi takke vår oppdragsgiver mnemonic AS og vår kontaktperson hos dem, Andreas Bråthen. Med et kritisk blikk og mye kunnskap har Bråthen gitt oss inspirasjon og utfordringer for å best mulig kunne besvare mnemonic sitt behov og ønske med tanke på rapportens intensjoner og mål. Videre vil vi takke vår veileder Erik Hjelmås ved Høgskolen i Gjøvik. Ved våre ukentlige - når alle var på Gjøvik og kunne i hvertfall - møter har vi fått luftet vår tanker, ideer og frustrasjoner, og blitt møtt med konstruktiv kritikk og oppløftende svar.

Til slutt vil begge gruppens medlemmer samlet takke hverandre for å ha vært en god samarbeidspartner. Uten noen uenigheter eller misnøye sammen med muligheten til å faglig kunne utfordre hverandre til å prestere best mulig, har dette prosjektet vært en spennende og minneverdig opplevelse. Til Lars, lykke til på mastergraden. Til Andreas, lykke til i arbeidslivet.

Andreas Moe(Oslo, 2014/05/16) Lars Christian Andersen (Gjøvik, 2014/05/16)

Innhold

Tabeller

Figurer

Ordliste

API Application Programming Interface. 8

C&C Command & Control. 10, 11, 15, 17

CDN Content Distribution Network. 11, 15

CLI Command-line interface. 20–22, 36, 44

CSS Cascading Style Sheets. 47

CSV Comma-separated values. viii, 9, 22–24, 31, 32, 35

DGA Domain Generation Algorithm. 3, 11, 14, 15, 17

DNS Domain Name System. xi, 1–4, 6–11, 18, 22–24, 29–31, 52

FA Flux Agent. 11

FF Fast Flux. 11, 51

FFM Fast Flux Monitor. 9

FFSN Fast-Flux Service Network. 9, 11, 15, 16

HiG Høgskolen i Gjøvik. 3–6, 35

HiOA Høgskolen i Oslo og Akershus. 5

HTML HyperText Markup Language. viii, 46, 47

HTTP Hypertext Transfer Protocol. 8

IDN Internationalised Domain Name. 14

IDS Intrusion Detection System. 7, 12

IKT Informasjons- og kommunikasjonsteknologi. 1, 22

IP Internet Protocol. 10, 11, 14, 23, 24, 41

IPS Intrusion Prevention System. 12

ISECLAB International Secure System Lab. 9

JSON JavaScript Object Notation. 8

NISlab Norwegian Information Security laboratory. 52

NorCERT Norwegian Computer Emergency Response Team. 35

NS Name Servers. 6

NSM Nasjonal Sikkerhetsmyndighet. 35

OS Operativsystem. 38

pDNS Passive Domain Name System. iv, viii, 1–4, 6–10, 13, 15, 18, 19, 21–24, 26, 28–33, 35, 39, 41–47, 49–52, 55

pDNS-DB pDNS Database. 2, 8, 9, 18, 26

POC Proof-Of-Concept. 2, 3, 33, 34, 51, 55

RR Resource Records. viii, 6, 7, 15

RRDNS Round-robin DNS. 11, 15

RS Recursive Server. 7, 8

SF Single Flux. 11

SOC Security Operations Center. 4

SVN Subversion. 35

TCP Transmission Control Protocol. 12

TLD Top Level Domain. vi, viii, 13–17, 31, 42, 43, 49, 51

TLP TrafikkLysProtokollen. 35

TTL Time To Live. vi, viii, 2, 11, 13–16, 21–23, 41–44, 49, 51

UML Unified Modelling Language. 27, 37

URL Uniform Resource Locator. 15, 42, 43

VPN Virtual Private Network. 11

Yapsy Yet Another Plugins SYstem. 39, 42

Definisjoner

Attribut En målbar egenskap som går igjen hos flere elementer

Cache Her ment som et område for midlertidig lagring av Domain Name System (DNS)-oppføringer

Cron-job Automatisert programvare som kjører ved bestemte intervaller

Daemon Program som kjører i bakgrunn uten brukerinteraksjon

Deteksjonsteknikker Her ment som en metode for å detektere ondsinnet aktivitet

Deque Double-ended Queue. Datastruktur hvor elementer kan hentes ut og legges til både foran og bak

Godsinnede domener Domener som ikke har onde hensikter

Kommunikasjonsteknikker Her brukt for teknikker som brukes for kommunikasjon mellom datamaskiner

Leksikalsk Egenskaper ved et ord

Nettverkspunkt Her brukt om fysiske lokasjoner hvor nettverkstrafikk passerer

Ondsinnet aktivitet Aktivitet med hensikt om å skade / ødelegge

Ondsinnede domener Domener som har onde hensikter og utfører ondsinnet aktivitet

Query Her brukt for en spørring utført for å få tak i DNS-informasjon

Reply Her brukt for et svar som inneholder DNS-informasjon

Resolver Klientens side av DNS-operasjoner. Ansvarlig for å hente ønskede ressurser fra DNS-servere

Resource Record Her brukt for et element som beskriver domener

Reverse Engineering Her brukt for prosessen ved å analysere kode for å forstå den oppbygging og logikk

Sensor Her brukt om en boks som analyserer passerende nettverkstrafikk og rapporterer til en sentral enhet

Server Her brukt om alle servere som behandler DNS-data

Skadevare Her brukt om programvare som utfører ondsinnet aktivitet

Stand-alone Her brukt om programvare som kjører uten påvirkning fra maskinvare og eller programvare

Svartelisting Her brukt om prosessen av å lage en liste med domener for sperring

Trusselaktør Her brukt om personene som står bak ondsinnet aktivitet

Trusselbildet Her brukt for å beskrive hvordan og hvilket trusler som finnes i det ville

Verbose Her brukt om det å skrive ut programmets gang mens det kjører

1 Introduksjon

1.1 Innledning

Kappløpet mellom sikkerhetsprofesjonelle og kriminelle samt andre trusselaktører kan sammenlignes mer med et maraton enn en sprint. Dette kappløpet er et maraton som aldri tar slutt, og de med ondsinnede intensjoner som regel er et eller flere steg foran. Sikkerhetsmiljøene prøver konstant å ta igjen de kriminelle men må gang på gang innse at ressurser og kunnskapen som finnes hos de forskjellige trusselaktørene kan være utrolig høyt[2]. Samtidig kan en trusselaktør også være en mindre sofistikert angriper, men alikevel ha skadepotensiale grunnet pågangsmot og ressurser. Samtidig har de med kriminelle eller de generelt ondsinnede intensjoner et forsprang. De spiller nemlig angriper, og sikkerhetsmiljøene forsvarer.

Informasjons- og kommunikasjonsteknologi (IKT) har iløpet av det siste tiåret blitt en svært sentral og dermed en mer og mer kritisk del av dagens samfunn. IKT-løsninger benyttes til alt fra kontroll av demninger og andre styringssystemer til nettbank og selvangivelse. Utfordringene som denne utstrakte avhengigheten og benyttelsen av IKT medfører må håndteres og tas seriøst på alle nivåer[3]. Ikke bare må disse utfordingene tas seriøst, men det må også kunne formidles godt nok av sikkerhetsmiljøene til personer med ansvar og eller avgjørelsesmyndighet, enten om det er på et lokalt, nasjonalt eller globalt nivå.

Aldri har cyberkriminalitet lønt seg så mye [4], og de kriminelle har dermed mer penger enn noen gang til å bruke på slike cyberoperasjoner mens sikkerhetsprofesjonelle kjemper en konstant kamp. På grunn av juridiske restriksjoner utkjempes denne kampen som regel i en defensiv posisjon hvor verktøykassen består av reaktive og prevantive sikringstiltak. Dette er også med på å videreføre problemet med at man ligger som regel et eller flere skritt bak potensielle angripere.

I denne rapporten skal det sees nærmere på en relativt ny teknologi fra 2005 kalt Passive Domain Name System (pDNS)[5]. Sammen med gruppens oppdragsgiver skal det identifiseres behovene til et rammeverk som kan brukes til analyse og deteksjon av sikkerhetshendelser ved bruk av data fra pDNS-sensorer som analysegrunnlag. I tillegg til - og som grunnlag for dette rammeverket - skal det kartlegges potensialet som finnes i et pDNS-datasett i form av forskjellige analyserbare attributer og elementer som kan benyttes til å skille ondsinnet fra godsinnet aktivitet. Målet er å belyse mulighetene som finnes innen denne teknologien og deretter kunne skape et utgangspunktet for et morderne, tilpassningsdyktig og attraktivt våpen for sikkerhetsprofesjonelle å ha med i sitt arsenal for behandling av sikkerhetshendelser.

1.2 Organisering

Denne rapporten er organisert i 8 forskjellige kapitler. I dette kapittelet, *Kapittel 1 - Introduksjon (s. 1)*, vil den tildelte oppgaven, gruppens bakgrunn og generell informasjon om oppgaven og rapportens struktur bli presentert. *Kapittel 2 - Bakgrunnsinformasjon (s. 6)* vil presentere en grunnleggende bakgrunn for Domain Name System (DNS), pDNS og

tidligere arbeid innenfor rapportens fokus, for at leser skal bedre kunne forstå de viktigste temaene i denne rapporten. Videre vil analyseteknikker og de vurderte analyserbare attributene ved pDNS-data som kan bli brukt i rammeverket bli forklart og presentert i *Kapittel 3 - Deteksjon og analyse (s. 10)*. I *Kapittel 4 - Kravspesifikasjon og design (s. 18)* presenteres et kravspesifikasjon- og designdokument for et rammeverk som skal kunne analysere pDNS-data med utgangspunkt i funnene gjort i *Kapittel 3 - Deteksjon og analyse (s. 10)*. Mot slutten, i *Kapittel 5 - Prototype (s. 33)* vil det bli utviklet og presentert en prototype til dette rammeverket. I *Kapittel 6 - Videre arbeid (s. 49)* vil det diskuteres rundt elementer av rapportens funn og rammeverkets design og implementasjon som kan forbedes. Det vil også gis forslag til endringer og alternative løsninger, samt områder som har mangler eller burde videreutvikles. Rapporten, prosjektet og prosjektarbeidet vil bli diskutert og kritisert i *Kapittel 7 - Diskusjon (s. 52)*. Til slutt vil det i *Kapittel 8 - Konklusjon (s. 55)* bli presentert en konklusjon med tanke på gruppens funn, resultater og refleksjoner.

1.3 Definisjon av oppgaven

Denne rapporten skal kartlegge teknikker som benyttes av skadevare og trusselaktører hvor man kan dra nytte av pDNS som datakilde for analyse og deteksjon av potensielt ondsinnet og eller mistenkelig aktivitet. Dette gjøres ved at den tilgjengelige dataen som finnes per DNS-resolve i en pDNS-loggfil skal analyseres for å avdekke potensialet for å kunne benytte de respektive attributtene - for eksempel Time To Live (TTL)-verdier - i vurdering om den gitte aktiviteten som analyseres er ondsinnet eller ikke. Den opparbeidede informasjonen og teorien rundt denne kartleggingen skal så strekkes ut til detaljanalyse av forskjellige egenskaper eller attributer ved pDNS-data som betegnes som analyserbare attributer. Denne kartleggingen og presenteringen av teknikker og analyserbare attributter skal legge grunnlaget for å utarbeide en overordnet kravspesifisering samt designdokument av et rammeverk som skal kunne detektere potensielt ondsinnet aktivitet basert på de kartlagte analyserbare attributene. Med utgangspunkt i den utviklede kravspesifikasjonen og designdokumentet ble det ønsket fra oppdragsgiver at det skulle bli utviklet en Proof-Of-Concept (POC) av rammeverket for å kunne presentere prosjektmedlemmenes tanker og avgjørelser med tanke på implementering av dette rammeverket, samt å kunne vise frem tiltenkt funksjonalitet og egenskaper ved rammeverket.

1.4 Avgrensninger

Denne rapporten skal ikke produsere et ferdig produkt som er klart til å kunne implementeres eller fullskalatesting i et produksjonsmiljø. Rapporten vil utarbeide en POC som er ment nettopp for å vise frem tiltenkt funksjonalitet så tett opp mot kravspesifikasjon og designdokumentet som mulig. Sluttproduktet og tyngden i denne rapporten vil være å presentere og diskutere de identifiserte og vurderte behovene til et rammeverk som er i stand til å detektere, analysere og varsle om mistenkelig og eller avvikende DNS-aktivitet. Det har med bistand fra oppdragsgiver blitt gjennomført en analyse av hvilke krav (som vist i *Kapittel 4 - Kravspesifikasjon og design (s. 18)*) som stilles til rammeverket. Det resulterende rammeverket skal ikke se på historiske søk i den forstand at det ser på sammenhengen mellom flere innlegg som ligger i en pDNS Database (pDNS-DB). Derimot skal rammeverket ta et innlegg som input av gangen og prosessere dette.

1.5 Formål

Formålet med denne rapporten er todelt. Først vil rapporten belyse og begrunne potensialet som et pDNS-datasett innehar med tanke på mulige indikatorer for deteksjon av ondsinnet aktivitet. Den andre delen av rapporten vil presentere et rammeverk - via en kravspesifisering, designdokument og en egen utviklet prototype - som er i stand til å analysere de kartlagte funnene av analyserbare attributer i pDNS-data. Disse to delene kombinert vil gi lesere av rapporten og spesielt rapportens oppdragsgiver et sterkt utgangspunkt for å vurdere nytteverdien sammen med en mulig implementasjon av et slikt rammeverk. Formålet med denne rapporten er å analysere, kartlegge og presentere hvordan oppgavens oppdragsgiver - mnemonic AS - kan implementere analyse av pDNS-data på en ny måte i samarbeid med sine eksisterende løsninger og potensielt som en selvstendig analyseenhet for andre virksomheter og eller personer.

1.6 Tidligere arbeid

Selv om pDNS er en relativt ny teknologi [5] har sikkerhetsmiljøet allerede produsert mye litteratur innenfor dette fagfeltet. Det har blitt produsert flere utgivelser som diskuterer og anbefaler forskjellige attributer som kan brukes i deteksjon av skadevare, eksempelvis Bilge et al.[6] og Holz et al.[7]. Det har også blitt utført forskning på Domain Generation Algorithm (DGA) av Yadav et al. [8]. Mer informasjon om tidligere arbeid finnes i *Avsnitt 2.3 - Tidligere eller lignende løsninger (s. 9)*. Prosjektets medlemmer hadde ved tildelingen av denne oppgaven ingen tidligere erfaring med pDNS-teknologien, og samtidig kun en hatt en teoretisk introduksjon til DNS.

1.7 Målgruppe

For denne rapporten er målgruppen tiltenkt å være analytikere tilknyttet oppdragsgivers virksomhet samt andre personer og eller organisasjoner med tilsvarende ønsker og mål for et rammeverk til analyse av pDNS-data. Videre er det også et interessefelt for personer med en faglig interesse rettet mot teorien knyttet til analyse av pDNS-data, herunder temaer som analyserbare attributer og videre arbeid med tanke på utvikling av terskeler, klassifiseringsalgoritmer (bedømme om noe potensielt er ondsinnet eller ikke) og nye anvendelser av rapportens funn. Denne rapporten skrives med et utgangspunkt om at leseren har en god beherskelse av de teknologier, protokoller og metodikker som rapportens funn baseres på, samt de teknikker som er benyttet i kravspesifisering og utvikling av prototypeprogramvare som en POC. Hvor det blir sett som nødvendig vil disse - eller våre eventuelle tilpassninger - bli beskrevet og det henvises til andre kilder for videre informasjon og eller utdypning.

1.8 Bakgrunn

Begge medlemmene av prosjektgruppen gjennomfører en bachelor i informasjonssikkerhet ved Høgskolen i Gjøvik (HiG). Denne utdannelsen med sin kombinasjon av data- og informatikkemner som programmering og databaseteori samt sikkerhetsrelaterte emner som sikkerhetsplanlegging og hendelseshåndtering har gitt denne gruppens medlemmer et godt utgangspunkt for å ha et godt læringsutbytte samt beherskelse av denne oppgaven. I løpet av utdannelsen ved HiG har prosjektets medlemmer gjennomgått emner som kan både knyttes tett opp mot denne rapportens innhold, men også andre emner hvor det har blitt lagt grunnlag for å beherske nytt materiale. Temaer som systemutvikling,

nettverk og kommunikasjon, og algoritmer er områder prosjektets medlemmer tror kan hjelpe oss iløpet av denne prosessen. pDNS er et relativt nytt område for oss begge noe som gjør dette til en spennende prosess med mye nytt å lære. For å kunne levere et godt resultat må gruppens deltagere sette seg inn i temaer som blant annet DNS, pDNS, botnet og deres kommunikasjonsteknikker og Python. Sett fra et prosjektstyringsperspektiv føler prosjektets medlemmer at det var et svært godt utgangspunkt med tanke på effektivitet, samarbeid og struktur. Prosjektets medlemmer har begge jobbet sammen i svært mange prosjekter i vår tid ved HiG. Dette har resultert i et god forsåelse av og kjennskap til hverandres styrker, svakheter og personlige rutiner.

Utenom studier er begge medlemmene av dette prosjektet svært intereserte i informatikk og bruk av data på hobbybasis. Interesseområder varierer fra oppsett og drift av servere til programmering og skripting i forskjellige programmeringsspråk. Ved siden av dette har Andreas Moe i gruppen et pågående arbeidsforhold med mnemonic AS hvor han er deltidsansatt som studentvakt / sikkerhetsanalytiker ved mnemonic sin Security Operations Center (SOC). Via dette arbeidet har Moe førstehåndserfaring med problematikken rundt benyttelse av pDNS-data i analyse av sikkerhetshendelser samt utfordringer og fordeler ved dagens verktøy.

1.9 Arbeidsform

Da dette er et prosjekt hvor en stor del av tiden vil gå med på datainnsamling og analyse av eksisterende faglitteratur ønsket prosjektgruppen å ivareta en smidig fremgangsmåte som enkelt kunne tilpasses nye funn og eller utfordringer som ble oppdaget underveis i prosjektet. I en prosjektgruppe med kun to medlemmer og et sterkt ønske om en flat og fleksibel struktur, følte prosjektets medlemmer at det ha en dynamisk prosess ville gi oss en bedre effektivitet. Med dette ønsket om en fleksibel struktur ble det dermed valgt å ta utgangspunkt i utviklingsmodellen Scrum[9] hvor det ville bli fjernet og endret på forskjellige elementer av metodikken for å skreddersy den til dette prosjekets behov og krav. For mange av elementene i den planlagte prosessen vil det være vanskelig å sette forventet nødvendig ressursbruk og en smidig modell som Scrum gjør at gruppen stadig kan gjøre justeringer.

Tanken bak å basere rapportens prosjektstyringsmetodikk på Scrum var som nevnt å kunne beholde en god oversikt over nåværende arbeidsoppgaver samtidig som metodikken var fleksibel nok til å imøtekomme og håndtere nye utfordringer eller krav som ble oppdaget underveis i prosjektets løp. Prosessen til rapporten har vært å dele arbeidet inn i Sprinter av 2 ukers varighet. Hver slik sprint starter med et planleggingsmøte. På disse møtene gjennomgås arbeidsoppgaver som ikke har blitt fullført, en vurdering av kommende periode sine mål og fordeling av arbeidsoppgaver. Sammen med dette produseres det en statusrapport som sendes til veileder, oppdragsgiver og publiseres på prosjektets hjemmeside. Disse to-ukers intervalene repeteres inntil prosjektets avsluttning som er satt til å være den 5. Juni 2014. Det vil kun opprettholdes loggføring og utarbeidelse av statusrapporter for de sprintene som inngår i tiden avsatt til selve rapporten. Dette vil da resultere i 8 statusrapporter med sluttdato ved innleveringsfrist den 19. Mai 2014.

1.10 Rammer

Prosjektet skal gjennomføres i tidsrommet 6. Januar 2014 til 5. Juni 2014. Innen for dette tidsrommet er frist for innlevering av sluttrapporten satt til 19. Mai og en presen-

tasjon skal avholdes 5. Juni. Det er da totalt satt av 19 uker til arbeid med oppgaven og produksjon av sluttrapporten, og 3 uker til resterende arbeid som en avsluttende presentasjon, egenevaluering, publiseringsavtale med mer. Med den avtalte tiden gitt i *Appendix D - Forprosjekt (s. 142)* på mellom 27.5 og 30 timer i uken har gruppen en forventet arbeidsinnsats på $\approx 1100 \pm 50$ timer. I *Appendix D - Forprosjekt (s. 142)* ble det utviklet en fremdriftsplanen som utgangspunkt for disponering i tiden tilgjengelig. Dette er representert via et Gantt-skjema. Et revidert skjema som viser forandringer til denne disponeringen er vist i *Avsnitt A.6 - Revidert Gantt-skjema (s. 65)*. Det ble ved prosjektets start ikke avdekket noe grunnlag for å etterspørre økonomisk stønad eller andre ressursmessige krav som for eksempel lagringsplass, prosesseringskraft, infrastruktur, ol. Dette medfører at alt arbeid med prosjektet utføres på medlemmenes personlige datamaskiner og eventuelt ressurser fritt tilgjengelig ved HiG. Prosjektorganisering vil foregå ved hjelp av daglig dialog og oppgavebehandlingssystemet Redbooth[1][2].

1.11 Øvrige roller

For dette prosjektet har gruppen fått tildelt Erik Hjelmås, Førsteamanuensis, Dr. scient. som veileder. Hjelmås er til daglig foreleser ved både HiG og Høgskolen i Oslo og Akershus (HiOA). Som veileder har Hjelmås bidratt med mye kunnskap og konstruktiv kritikk både av det faglige men også tematikk som omhandler rapportskriving.

Oppdragsgiver for denne oppgaven er som nevnt sikkerhetsfirmaet mnemonic AS. Som kontaktperson for prosjektets oppdragsgiver har gruppen blitt tildelt Andreas Bråthen. Bråthen er tidligere student ved Høgskolen i Gjøvik og har i dag stillingen som sikkerhetsanalytiker / konsulent hos mnemonic. Han har tidlig i prosjektet vist stor interesse for å dele kunnskap og tips, og gruppen ser på han som en stor ressurs med mye å bidra til i denne oppgaven.

[1] `http://www.redbooth.com`
[2] Tidligere kalt Teambox, endret navn 21. Jan 2014

2 Bakgrunnsinformasjon

I de kommende kapitlene i denne rapporten vil temaer som DNS og pDNS bli diskutert. Prosjektgruppens rapporten sikter som tidligere nevnt inn på en målgruppe som allerede vil ha en forståelse for mange av teknologiene og temaene som vil bli ta opp. Selv om dette er tilfellet vil det bli forklart litt rundt de forskjellige temaene for å bekrefte at både skribenter og lesere er inneforstått med hva denne rapporten mener ved forskjellige ord, utrykk og sentral tematikk som for eksempel pDNS.

2.1 Domain Name System

DNS fungerer kort fortalt som internettets telefonbok. Den spiller en meget stor rolle i dagens kommunikasjon over internett ved at den oversetter fra et navn som er enkelt å huske (f.eks www.example.org) til en IP-addresse som er forståelig for enheten som forespør kommunikasjonen [10]. En oversettelse kan være fra *www.example.org* til 93.184.216.119 via Google Public DNS name server (8.8.8.8).[1]

DNS består av tre hoveddeler, *Domain Name Space* og *Resource Records (RR)*, *Name Servers (NS)*, og *Resolvers*[11]. Domain Name Space består av et tre av domenenavn hvor hver node har null eller flere RR. Disse RR-ene inneholder informasjon om data assossiert med hvert domenenavn. NS inneholder informasjon om deler av treet, og informasjon om andre NS. Resolvers er klientens måte å få tilgang til NS og hendelsesløpet er at klienten sender en *query* til en NS og får deretter en RR tilbake. Ved de tilfeller hvor RR inneholder den forespurte informasjonen kalles det en *non-recursive query*. Hvis derimot NS ikke har informasjon om det forespurte domene vil den returnere en annen NS og klienten må deretter gjenta forespørselen. Dette kalles *recursive query*[12].

Domain Name Space kan deles inn i noe kalt soner, og ved å dele det opp på denne måten kan ansvaret fordeles utover[12]. Dette kan forklares godt ved å se på *no*-domenet. Dette er delt inn i forskjellige soner, blant annet *hig.no* og *vg.no*. Her delegeres ansvaret til henholdsvis HiG og VG. Soner inneholder ingen noder i de delegerte subdomenene, noe domener gjør, så NS laster inn soner istedet for domener for å holde informasjonsmengden nede. Domener kan ofte inneholde mer informasjon enn NS trenger.

2.2 Passive Domain Name System

pDNS er en relativt ny teknologi, med den første konkrete publiseringen om metodikken gitt ut av Florian Weimer i 2005[5]. I følge Weimer[5] betegnes pDNS som:

> "... er prosessen av å lagre DNS-queries og eller DNS-replies, og å bruke disse dataene til å konstruere delvise replikater av så mange DNS-soner som mulig"

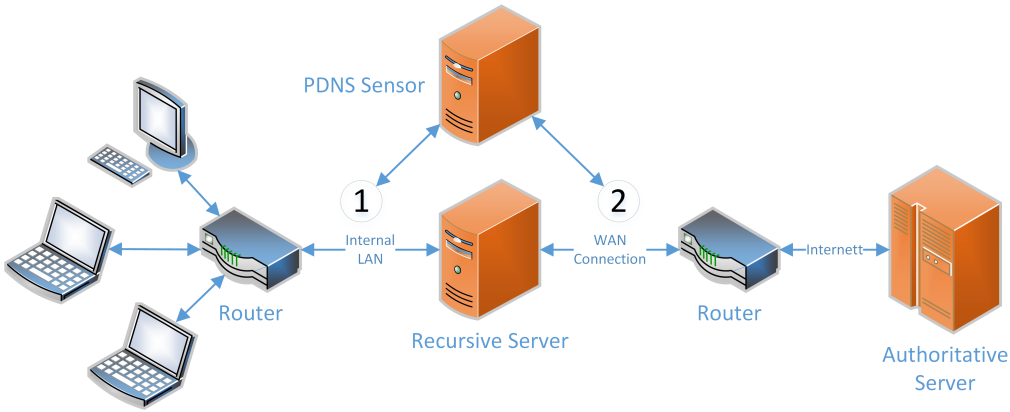
Med dette grunnlaget er det mulig å si at tanken bak konseptet med pDNS er å overvåke et eller flere nettverkspunkter hvor det flyter større mengder med DNS-trafikk. Dette kan gjøres enten separat, eller samlet hvor resultatene aggregeres i en sentralisert database. Deretter lagres DNS-queries eller DNS-replies i en database. Dette fører til at man vil

[1] Resultat av `dig www.example.org@8.8.8.8` gjennomført 22. Apr. 2014

kunne bygge opp en replikat av DNS-data og DNS-soner med oversikt over når, hvor og hva som ble forespurt og eller gitt som svar. Å kunne konstruere en slik database har mange fordeler. Hvis trusselaktører for eksempel fjerner eller endrer et domene sitt RR vil man ha problemer med etterforskning og sporbarhet av ondsinnet aktivitet. Et godt eksempel her er at en analytiker korrelerer flere potensielle sikkerhetshendelser opp mot hverandre og avdekker ondsinnet aktivitet opp mot et eller flere mistenklige domenenavn som foregikk noen uker siden, men er ikke i stand til å etterforske noe videre siden de respektive domenenavnene sine RR har blitt endret. I tillegg til å kunne skape en kronologisk *backup* av DNS-hendelser på nettverket vil det også tilby en konstant strøm av *live* DNS-aktivitet, selv om dette også er noe som kan fanges opp av andre enheter som brannmurer eller for eksempel Intrusion Detection System (IDS)-løsninger.

Sensorplassering

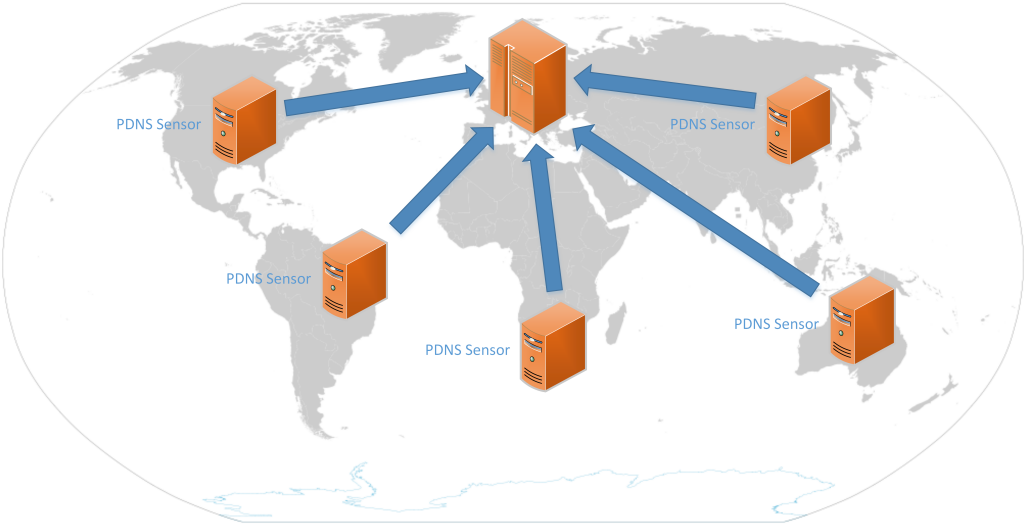
Når man skal lytte til nettverkstrafikk med intensjon om å fange opp pDNS-data så er det viktig hvor man plasserer ut en eventuell pDNS-sensor. I *Figur 1 - Plassering av pDNS-sensor (s.7)* ser man en overordnet og generisk topologisk oversikt over et nettverk med to potensielle plasseringer av en pDNS-sensor, punktene 1 og 2. Forskjellen mellom de forskjellige plasseringene her er henholdsvis om sensoren skal plasseres slik at den detekterer DNS-queries før den kommer til en *Recursive Server (RS)*, eller etter at en RS har mottatt en query og må videre i DNS-hierarkiet for å kunne besvare den forespurte DNS-querien.



Figur 1: Plassering av pDNS-sensor

Ved å plassere en pDNS-sensor på punkt 1 i *Figur 1 - Plassering av pDNS-sensor (s.7)* vil sensoren ligge i det lokale nettverket og detektere alle DNS-queriene før de når en RS. Dette gjør at en slik sensor vil få med seg alle DNS-queriene som blir gjort, uansett om de er nye, unike eller allerede i en potensiell *cache*. Andre fordeler ved å ha en sensor plassert ut på et slikt punkt vil være at man kan observere og registrere de individuelle klientene som har forespurt visse domener, noe som øker verdien til datasettet når det kommer til sporbarhet og fremtidig analyse av sikkerhetshendelser. Ved å plassere en pDNS-sensor på et slikt punkt vil man potensielt få store mengder med data ettersom DNS er en svært hyppig brukt protokoll på de fleste nettverk. Dette kan potensielt være et problem med tanke på ressurser og kapasitet til å behandle og registrere alle de observerte DNS-queriene.

En annen mulighet når det gjelder plassering av en pDNS-sensor er som vist i punkt 2 i *Figur 1 - Plassering av pDNS-sensor (s.7)*. Ved å plassere en sensor ved dette punktet så vil kun de DNS-queriene som ikke ligger i en potensiell cache bli fanget opp. Dette vil si at kun nye og sjeldent besøkte domenenavn vil bli fanget opp av en sensor plassert på et slikt punkt. I forhold til en plassering som punkt 1 vil dette punktet føre til en mye lavere mengde data som sensoren må prosessere ettersom mye av hovedtrafikken med tanke på DNS-queries vil være lagret i en cache. Ved å plassere en sensor på et slikt punkt vil man miste evnen til å bygge opp statestikk over total DNS-aktivitet på et nettverk. Man vil heller ikke kunne lagre hvilken klient som har sendt de respektive DNS-queriene ettersom det da er en RS som gjennomfører videre forespørsler om DNS-replies i DNS-hirarkiet. I *Figur 2 - Internasjonal spredning av pDNS-sensorer (s.8)* vises et eksempel på pDNS-sensorer på en større skala. Her aggregeres dataen som blir fanget opp i en samlet database.



Figur 2: Internasjonal spredning av pDNS-sensorer

pDNS-produkter og tjenester

Selv om pDNS er som tidligere nevnt en relativt ny teknologi[5] finnes det en rekke forskjellige implementasjoner og egenutviklede løsninger der man har sett behov for en slik datainnsamling. Disse implementasjonene varierer fra tjenester man finner som et Application Programming Interface (API) på internett til *stand-alone*-programmer skrevet i programmeringsspråket C og lignende. Virustotal[2] leverer primært en tjeneste for skanning og vurdering av nettsider og potensielt skadelige filer. En tjeneste de også leverer er søking i en pDNS-DB samt et API for interaksjon med denne tjenesten via Hypertext Transfer Protocol (HTTP)-POST forespørsler med JavaScript Object Notation (JSON)-objekter som svar. En lignende tjeneste for søking i en pDNS-DB leveres også fra BFK edv-consulting[3].

Hvis man ønsker å sette opp sin egen pDNS-sensor finnes det mange muligheter og

[2] `http://www.virustotal.com` - et datterfirma av Google
[3] `http://www.bfk.de/bfk_dnslogger_en.html`

potensielle løsninger. En fremtredende implementasjon er programvaren 'PassiveDNS' laget av Edward Bjarte Fjellskål[4] som har blitt publisert på nettsiden Github for snart tre år siden. Dette produktet kommer rapporten tilbake til i *Avsnitt 4.1.4 - Comma-separated values (CSV)-format - Input (s. 22)* ettersom dette er et av produktene prosjektets oppdragsgiver har benyttet til innsamling av pDNS-data. Videre så er det fullt mulig å lage sine egne løsninger som benytter flere verktøy sammen for å opparbeide seg en pDNS-DB. Dette kan for eksempelvis gjøres ved å fange data med verktøyet tcpdump[5], samt utvikle egne løsninger for lagring, søking og lignende.

2.3 Tidligere eller lignende løsninger

Som nevnt i *Avsnitt 1.6 - Tidligere arbeid (s. 3)* så har fagmiljøene produsert en del litteratur om pDNS allerede. I de ni årene som har gått siden pDNS ble utviklet, har både teoretiske og praktiske fremskritt blitt gjort innen dette temaet. Av dette så er det noen systemer som har likheter med PalmaDNS, og Exposure, utviklet av Bilge et al.[6], er et slikt system. Exposure er et system for å detektere ondsinnede domener ved hjelp av passiv analyse av DNS, og bruker til sammen 15 forskjellige attributer til deteksjonen. Tester viser at systemet er skalerbart og fungerer godt i produksjon. Systemet er idag åpent for bruk av publikum på nettsiden til International Secure System Lab (ISECLAB)[6].

Caglayan et al.[13] presenterte i 2009 en Fast Flux Monitor (FFM) som kunne detektere Fast-Flux Service Network (FFSN) ved hjelp av aktiv og passiv DNS-monitorering. Dette systemet kan detektere FFSN i sanntid. Empiriske tester ga en deteksjonsnøyaktighet på 96.6%. Se *Avsnitt 3.1.2 - Fast-Flux Service Network (s. 11)* for informasjon angående FFSN. Perdisci et al.[14] utviklet i 2012 et system kalt FluxBuster som kun fokuserer på deteksjon av FFSN. Også her blir det fastsatt visse attributer som brukes i deteksjon. Tester viser at FluxBuster er i stand til å detektere tidligere ukjente FFSN før de kommer i offentlige svartelister.

Utover dette så eksisterer det idag flere generelle deteksjonssystemer. Velkjente Snort[7] er et slikt system. The Bro Network Security Monitor[8] er et annet som skiller seg litt ut i mengden av deteksjonssystemer. Dette er systemer som ikke kun fokuserer på pDNS, men kan utføre deteksjon over hele spekteret. Man kan se for seg at når PalmaDNS implementeres med eksisterende moduler hos mnemonic så vil hele dette systemet kunne sammenlignes med Snort og The Bro Network Security Monitor.

[4] `https://github.com/gamelinux/passivedns`
[5] http://www.tcpdump.org/
[6] http://exposure.iseclab.org/
[7] http://www.snort.org
[8] http://www.bro.org

3 Deteksjon og analyse

3.1 Introduksjon

3.1.1 Mål

Et av hovedmålene ved denne rapporten er å kartlegge og identifisere behovene til et rammeverk som skal kunne analysere et pDNS-datasett, samt detektere og varsle om potensielt ondsinnet aktivitet. For å kunne detektere og vurdere om en gitt aktivitet er ondsinnet, må det først identifiseres hva det er som skal vurderes og benyttes i en analyse. Dette målet legger grunnlaget for det kommende kapittelet som omhandler analyse og kartlegging av trusselbildet og teknikker benyttet av skadevare eller generell aktivitet med ondsinnede intensjoner. Etter at dette er utarbeidet tar dette kapittelet for seg å beskrive de forskjellige attributene og elementene som finnes i et pDNS-datasett som kan benyttes i å analysere og potensielt avdekke de vurderte skadevareteknikkene.

Målet for dette kapittelet er å undersøke og analysere datasettet som gis fra pDNS-sensorer og som finnes i DNS-trafikk. Denne analysen kombinert med en kartlegging av fremtredende skadevareteknikker vil danne grunnlaget for å kunne identifisere behovene til et rammeverk som skal gjennomføre en slik analyse. For mer informasjon om dette rammeverket henvises det til *Kapittel 4 - Kravspesifikasjon og design (s. 18)*. Samtidig vil den presenterte analysen kunne benyttes i andre løsninger eller forskning og innehar dermed egen nytteverdi, seperat fra denne rapportens rammeverk. Ved å bruke tidligere funn og forskning presenteres dette på en kortfattet måte mens det fortsatt beholder sin integritet.

3.1.2 Skadevareteknikker

For å skjule sin aktivitet og forbli usett på en infisert klient mens den kommuniserer over et nettverk benytter dagens skadevare et stort utvalg av teknikker. Disse teknikkene er ment for å øke levetiden til et infisert system, da altså hvor lenge og i hvilket omfang en inntrenger har tilgang til å utføre sine ondsinnede handlinger. I dette avsnittet presenteres kjente kommunikasjons- og skadevareteknikker som potensielt kan detekteres ved hjelp av data og analysen av denne som finnes i et pDNS-datasett. Egenskapene og den generelle funksjonaliteten og intensjonen til disse teknikkene vil bli diskutert. Senere i rapporten vil attributer som kan være nyttig i deteksjon av disse bli presentert og vil bygge på disse avsnittene.

Statisk domenenavn

Et botnet er en samling av infiserte maskiner som styres av en felles eier. I et botnet må de infiserte maskinene ta kontakt med en server kalt Command & Control (C&C)-server for å få videre instruksjoner etter infisering. Den som eier og drifter en slik C&C-serveren kalles en botmaster. Den enkleste teknikken for at klientene skal finne C&C-serveren er å inkludere en statisk liste over Internet Protocol (IP)-er i skadevaren som den prøver å koble seg til[15]. Når sikkerhetseksperter utfører *Reverse Engineering* er det mulig å få tak i dette innholdet og blokkere eller etablere en juridisk prossess for å deaktivere de respektive serverene. Videre har man kunne etablere er mer fleksibel infrastruktur ved å

benytte domenenavn kontra IP-er i skadevaren. Dette vil da føre til at skadevaren har en liste over domenenavn og utfører DNS-queries for å finne IP-en til sin C&C-server. Ved en slik metode kan en botmaster endre hva dette domenenavnet skal resolve til hvis det er behov for dette, og dermed lettere kunne ivareta sin infrastruktur.

Som nevnt er det mulig gjennom reverse engineering og skadevareanalyse å finne disse domenenavnene og statiske IP-er. En annen teknikk for dette er ved å detektere mistenklige DNS-queries, enten om dette oppdages via omdømme kilder, tilfeldighet eller at et periodisk mønster blir avdukket. Ved slike anledninger har sikkerhetspersonell mulighet til å blokkere eller fortsette målrettet overvåkning av situasjonen.

Fast-Flux Service Network

For at legitime tjenester skal kunne tilby høy tilgjengelighet, ytelse og skaleringsmuligheter brukes en teknikk kalt Content Distribution Network (CDN). Et CDN fungerer ved å spre noder over flere lokasjoner, og ved en etterspørsel vil brukeren få tilbudt den etterspurte tjenesten fra den noden som leverer tjenesten raskest. Her blir det en balanse mellom avstand og ytelse på nodene. En annen teknikk for å fordele last ved hjelp av DNS er Round-robin DNS (RRDNS)[16]. Dette er en lastbalanseringsteknikk hvor etterspørseler blir levert til forskjellige noder etter en algoritme for å fordele lasten mellom nodene. FFSN bygger på teknikkene CDN og RRDNS, og misbruker dette til å kunne opprettholde en høy tilgjengelighet, ytelse, skaleringsmuligheter og motstandsstyrke mot svartelisting for sine botnet.

I et FFSN bytter koblingen mellom IP-adresser og domener hyppig[17], og denne teknikken gjør det vanskeligere å blokkere og stenge ned C&C-servere. Teknikken kan deles inn i to kategorier; Single Flux (SF) og Fast Flux (FF). I et SF vil et domenenavn peke til forskjellige IP-er ettersom når en DNS-spørring sendes[17]. Med dette menes det at når TTL har gått ut på de eksisterende rutene vil klienten motta en ny IP til en annen Flux Agent (FA). Kommunikasjonen mellom klient og C&C-server vil dermed gå igjennom forskjellige FA for å gjøre deteksjon vanskeligere. FA fungerer som en proxy. Et FF bygger på det samme prinsippet som et SF, men endrer også IP-adressene på DNS-serverene[7].

Domain Generation Algorithm (DGA)

En annen teknikk som trusselaktører kan benytte seg av til å unngå deteksjon samtidig som de opprettholder en redundant infrastruktur er DGA. DGA er en teknikk hvor skadevaren vil benytte seg av en algoritme som ut i fra for eksempel tid, dato eller lignende flytende variabler og en initieringsverdi[17], genererer et domenenavn. Botmasteren bruker samme algoritme og verdi, og registrerer deretter et domene med det genererte navnet. Kjent skadevare som *Conficker*[18] og *Murofet*[19] genererer en større liste med domenenavn og tester alle disse. Selv om sikkerhetseksperter finner ut hvordan algoritmen fungerer ved hjelp av *Reverse Engineering* så vil listen være for stor til at det er mulig å registrere domenene før de blir registrert av botmasteren. Slike generert domenenavn vil som regel for selv et utrent øye se mistenkelig ut.

3.1.3 Deteksjonsteknikker

For å kunne detektere potensielt ondsinnet aktivitet på et nettverk er det forskjellige metoder for dette. Ved nettverk som allerede innehar mye nettverksutstyr som for eksempel Virtual Private Network (VPN)-gateway, brannmurer, rutere og lignende vil analyse av loggfiler fra disse komponentene være svært nyttige. Videre kan man utvide nettverks-

infrastrukturen med spesielle enheter og programvare nettopp for dette. Eksempler på dette er IDS eller Intrusion Prevention System (IPS) som f.eks Snort[1], Suricata[2] eller FireEye NX-serien[3].

For at et IDS-system eller lignende analysesystem for deteksjon av potensielt ondsinnet aktivitet skal fungere godt må det ha en høy andel riktige alarmer, og en lav andel falske alarmer. Dette kalles henholdsvis *true-positive* og *false-positive*[20]. Når skadevare skal detekteres så er det to forskjellige fremgangsmåter som er mye brukt i IDS-systemer[21].

Signatur

En av metodene som benyttes i IDS-løsninger er å strukturere deteksjonen av ondsinnet aktivtet basert på ferdig utarbeidede regler, såkalte signaturer. Signaturbaserte deteksjonssystemer bygger på at det lages sett med regler som brukes for å detektere uønsket oppførsel ved å se om en gitt pakke som sendes over nettverket og forbi den respektive sensoren stemmer overens med en signatur i dens database. Får man et slikt treff vil IDS-sensoren alarmere i henhold til sin konfigurasjon. Eksempel på et slikt system er Snort[4]. *Figur 3 - Enkel Snort-regel (s.12)* viser en enkel Snort-regel presentert i Roesch[22]. Denne regelen logger all Transmission Control Protocol (TCP)-trafikk som går inn på port 79 til adresser i 10.1.1 klasse C. Signaturbaserte deteksjonssystemer har mange fordeler, deriblant enkelhet, lav andel falske alarmer og nøyaktig deteksjon. På grunn av sin signaturbaserte fremgangsmåte så må skadevaren være kjent for at systemet skal kunne detektere det, og det er hyppigheten og kvaliteten på nyutviklede signaturer som differensierer deteksjonssystemene fra hverandre.

```
log tcp any any -> 10.1.1.0/24 79
```

Figur 3: Enkel Snort-regel

Statistisk anomali

En annen metode for å avdekke potensielt uønsket trafikk på nettverket kan være IDS/IPS-systemer som baserer sin metodikk for deteksjon på statistisk anomali fremfor signaturer som beskrevet i delavsnittet over. Statistisk anomali-baserte deteksjonssystemer bygger opp en database med normal og godkjent - en såkalt baseline som må etableres over tid - data og bruker deretter denne til å detektere avvik. Det defineres profiler for normal oppførsel og alt som avviker fra dette klassifiseres som ondsinnet[23]. Et eksempel på en statistiske teorem er *Bayes Teorem*[24] som kan brukes til å detektere unormal aktivitet. En fordel som anomali-basert deteksjon har over for eksempel signatur-baserte systemer er at de er i stand til å kunne oppdage nye og tidligere ukjent sårbarheter[25], såkalte Zero-Day sårbarheter. Ulempene med et slikt system vil være at det kan medføre svært mange falske positive alarmer[26]. Dette kan skyldes at den utviklede baselinen er for snever i forhold til avvikende tidsperioder. Et godt eksempel her er statisk anomalibaserte IDS/IPS-er som produserer svært mange alarmer i høytider og fellesferien ettersom dette avviker fra normal nettverksaktivitet.

[1] http://www.snort.org/
[2] http://suricata-ids.org/
[3] http://www.fireeye.com/products-and-solutions/web-security.html
[4] http://www.snort.org/

3.2 Attributanalyse

Når pDNS-data skal analyseres må det være fastsatte elementer som skal testes. I dette kapittelet vil *Avsnitt 3.2.1 - Attributoversikt (s. 13)* presentere attributer som potensielt har en nytteverdi med tanke på deteksjon og videre analyse av ondsinnede domenenavn. Disse attributene vil være elementer som alle innleggene i pDNS-databasen kan testes på. Eksempler på et attribut kan være lengden på et domenenavn, eller rett og slett verdien til TTL. Bilge et al.[6] presenterer en inndeling og kategorisering av analyserbare attributer. Denne rapporten og sin analyse av attibuter vil videreføre to av disse kategoriene for å bedre kunne skape en oversiktlig struktur. Kategoriene for attributer er: *navnbaserte* attributer og *svarbaserte* attributer samt en oppsamlende kategori kalt *diverse* for attributer som ikke faller innenfor de to andre kategoriene. En presentasjon av disse samt tilhørende forklaringer finnes i *Tabell 1 - Kategorier av attributer (s. 13)*.

Attributkategorier	
Navnbaserte	Tar for seg analyserbare elementer som har med det etterspurte domenenavnet å gjøre. Eksempler her vil være fordeling av bokstavfrekvenser eller antall numeriske tegn.
Svarbaserte	Tar for seg attributer og elementer som ikke nødvendigvis er levert sammen med en DNS-reply men er avhengige av etterspørring av ytterlig informasjon fra eksterne kilder. Eksempler på dette kan være å sjekke hvor lenge et domene har eksistert via WHOIS created-at eller sammenheng mellom Top Level Domain (TLD) og serverens geografiske plassering.
Diverse	Dette er en oppsamlende kategori av attributer og elementer prosjektets medlemmer ikke mener direkte passer inn i noen av de andre kategoriene. Målet er å holde denne gruppen minst mulig.

Tabell 1: Kategorier av attributer

3.2.1 Attributoversikt

I *Tabell 2 - Attributer (s. 14)* presenteres en liste over attributer som potensielt kan fungere til å skille ondsinnede og godsinnede domener. Dette er en aggregert liste med attributer tatt fra tidligere forskning innen pDNS[27, 28, 6]. Dette er ikke en uttømmende oversikt over alle potensielle attributer, men et utvalg og en pekepinne for videre utvikling og forskning. Nye teknikker som skadevare kan misbruke kan komme senere, og det er viktig å ikke bli fastlåst i de attributene som er nevnt i denne rapporten. Trussellandskapet endrer seg, og det er viktig å gjenspeile dette i attributer som blir brukt i deteksjonen. Rammeverket skal dermed basere seg på at det også skal være andre attributer som kan brukes. Et dypdykk ned i alle disse attributene ville vært meget omfattende og det har dermed blitt tatt et valg om å kun fokusere på et utdrag av fem attributer. Utdraget er basert på en blanding av nytteverdi, enkelthet og tilgjengelighet av data. Ved å presentere disse fem attributene er det mulig å vise nytteverdien til disse i et slikt rammeverk som er omtalt. De uthevede feltene i *Tabell 2 - Attributer (s. 14)* presenterer de

fem fokusområdene.

Attributer	
Navnbaserte	(1) Antall numeriske tegn
	(2) Lengde på lengste meningsfulle streng
	(3) Tegnfrekvens
	(4) Entropi
	(5) Lengde på domenenavn
	(6) Internationalised Domain Name (IDN)-homografi
Svarbaserte	(7) TTL-verdi
	(8) TLD
	(9) Antall forskjellige IP-adresser
	(10) Dato for opprettelse av domene
Diverse	(11) Domeneomdømme
	(12) IP-adresseomdømme

Tabell 2: Attributer

3.2.2 Analyserte attributer

(1) Antall numeriske tegn

IDN bruker *ASCII* for å representere domenenavn[29]. I den senere tid har det derimot blitt introdusert en teknikk som bruker *Punycode* til å skape et IDN. *Punycode* gjør om kodingen fra *Unicode* til *ASCII*, noe som gjør at *Unicode* også kan brukes i domenenavn[30]. Domenenavn har dermed et bredt utvalg av tegn som kan benyttes, hvor noen tegn kan gå igjen oftere.

En opptelling av domener med numeriske tegn i topp-besøkte domenelister fra åpne kilder [31, 32] viser en veldig lav andel av domenenavn som inneholder numeriske tegn (Alexa: 0.072%, Moz: 0.028%). Disse domenelistene gjelder da for domener med gode intensjoner og gir en indikasjon på bruken av numeriske tegn i godsinnede domener. En annen undersøkelse gjort av [33] viser derimot at rett over 9% av deres undersøkte .com domener(totalt 102,815,927) inneholder numeriske tegn. Dette er en undersøkelse basert på både godsinnede og ondsinnede domener. På den andre siden så har undersøkelser av ondsinnede domener hentet fra malwaredomainlist.com[34] vist innhold av numeriske tegn på rundt 0.199%. Selv om dette er en svært lav andel er det ≈ 2.7 ganger mer numeriske tegn enn funnet i Alexa[31] og ≈ 7.1 ganger mer i Moz[32].

Tidligere forskning har gjentatte ganger brukt antall numeriske tegn som et attributt for deteksjon av skadevare. Under utviklingen av Exposure vurderte Bilge et al.[6] at antall numeriske tegn kunne være en indikasjon på et DGA-generert domenenavn. Stalmans & Irwin[35] valgte å inkludere tegnfrekvens i sitt rammeverk. I deres analyse kommer det godt frem at det er en klar forskjell mellom godsinnede og ondsinnede domener når det gjelder numeriske tegn. Dette bekreftes av Yadab & Reddy[36] som sier at entropien, bokstavkombinasjonen, for ondsinnede domenenavn er merkant forskjellig fra godsinnede domenenavn.

Ettersom numeriske tegn forekommer svært sjeldent i domenenavn kan dette være en potensiell indikator på uregelmessig aktivitet. Domener med mange numeriske tegn vil skille seg svært ut i mengden av trafikk. For at dette attributet skal kunne brukes i analysen må det statistisk regnes ut en terskel.

(5) Lengde på domenenavn

Som nevnt tidligere så kan DGA brukes til å generere domenenavn. Det er lite forskning innen pDNS som har brukt lengde på domenenavn som et attribut til deteksjon, men det betyr ikke at det er ikke er muligheter for at dette attributet fortsatt skal kunne brukes i deteksjon av skadevare.

Phishing er en teknikk som går ut på å sende brukerene til ondsinnede sider som utgir seg for å være legitime sider[37]. Dette er gjerne sider hvor sensitiv informasjon skrives inn. McGrath & Gupta[38] sine funn etter analyse av domener brukt til phishing viser at det var en markant forskjell mellom disse domenenavnene og godsinnede domenenavn på internett. Datagenetics[33] presenterer en oversikt over lengde på domenenavn innenfor TLD-ene .com og .net, og det kommer godt frem at det er visse lengder som er mer vanlig enn andre. Det faktumet at godsinnede domener har en større sannsynlighet for å ha visse lengder gjør at det er mulig å se på avvik og dermed detektere potensielle ondsinnede domener.

Ma et al.[39] støtter dette ved å inkludere lengde på domenenavn som et av attributene i sitt system for klassifisering av Uniform Resource Locator (URL)-er. Ved å bruke de tre leksikalske attributene lengde på domenenavn, lengde på URL og antall punktum til å trene opp klassifisereren oppnår de en feilrate mellom 1.9% og 3.5% i sine tester. I utvikling av et annet deteksjonssystem for ondsinnede URL-er gjort av Chong et al.[40] konkluderer de også med at leksikalske attributer kan brukes til å trene opp deteksjonssystemer.

Antonakakis et al.[41] presenterer et interessant resultat i sin testing av deres nyutviklede deteksjonssystem for C&C, Pleiades. Systemet detekterte en DGA som senere ble tilknyttet Zeus.v3. Det interessante er at analysen av DGA-en viser at den genererer domenenavn med lengde mellom 33 og 45 alfanumeriske tegn. Slike analyser kan i etterkant gi en terskel for kjente DGA-er, og muligheten for å bruke lengden av domenenavn som et attribut for å detektere allerede kjente DGA-er er der.

(7) TTL-verdi

Når et RR blir returnert vil den inneholde blandt annet en TTL-verdi. Denne verdien forteller hvor lenge en *Resolver* skal ta vare på den respektive RR før den slettes, dvs hvor lenge den skal ligge i *cachen*. *Tabell 3 - Oppbygging av RR (s. 15)* viser hvordan et RR er bygd opp [10].

Name
Type
Class
TTL
RDLENGTH
RDATA

Tabell 3: Oppbygging av RR

Holz et al.[7] observerte at domener i et FFSN setter TTL-verdier som følger visse karakteristikker. De så at TTL-verdien til ondsinnede domener, som var 600 sekunder, var lavere enn godsinnede domener. I deres publikasjon ble det trukket linjer mellom FFSN og CDN når det kommer til oppførsel. Grunnet dette anser de ikke TTL-verdi som et godt attribut alene da det ikke er mulig å skille mellom FFSN og CDN. TTL kan derimot brukes til å skille FFSN og CDN fra RRDNS. Bilge et al.[6] bekrefter nytteverdien til TTL-verdier i deres utvikling av Exposure noen år senere hvor de valgte å inkludere TTL-verdi som en potensiell faktor for deteksjon av ondsinnede domener. Deres empiriske analyse av TTL-verdier konkluderer også med at TTL settes lavt for ondsinnede domener, og de observerte spesielt en høyere andel i spekteret *[0, 100]*. Tankegangen om at domener i

et FFSN karakteriseres med lav TTL er noe som går igjen i flere arbeid, noe Perdisci et al.[14] viser i sin liste over potensielle attributer. For at dette attributet skal være effektivt i deteksjon av ondsinnede domener må en fornuftig terskel settes. Her er det mulig å se tilbake på tidligere forskning[7, 6] og bruke tall de har observert i sine FFSN.

(8)TLD

For at en nettside skal kunne ha et domenenavn må den registreres under en TLD. Eksempler på TLD-er er *.com*, *.net* og *.info*. Registreringen hos disse TLD-ene styres av forskjellige organer rundt om i verden, og registreringsprosessen inneholder ikke alltid like mye arbeid for registranten. Ikke alle domener som blir registrert er godsinnede, og noen TLD-er kan ha høyere andel ondsinnede domener en andre. Når TLD-er skal rangeres så er det mulig å vekte verdiene for å oppnå et mer realistisk resultat. McAfee sin trendrapport[1] fra 2011 presenterer to forskjellige teknikker for å sette risikoraten til TLD-er, vektet og uvektet risiko. I den vektete risikoen kommer 50% av verdien fra forholdet mellom antall ondsinnede domenene i en TLD og alle domenene i den samme TLD-en, og 50% av verdien fra forholdet mellom antall ondsinnede domene i en TLD og alle domener for alle TLD-er totalt. Uvektet risiko derimot ser kun på forholdet mellom antall ondsinnede domener i en TLD og alle domenene i den samme TLD-en.

Country or Name	**Region**	**TLD**	**2010 Worldwide Risk Rank**	**2010 Weighted Risk Ratio**	**2010 Unweighted Risk Ratio**
Commercial	Generic	COM	1	31.3%	6.1%
Information	Generic	INFO	2	30.7%	46.6%
Vietnam	APAC	VN	3	29.4%	58.0%
Cameroon	EMEA	CM	4	22.2%	44.2%

Tabell 4: Utsnitt av tabell over risikorangering av TLD-er[1]

I *Tabell 4 - Utsnitt av tabell over risikorangering av TLD-er[1] (s. 16)* er de fire TLD-ene som fikk høyest vektet risikorate av McAfee i 2010. Trussellandskapet er i konstant forandring og hvilket TLD-er som blir brukt til ondsinnede hensikter er også i forandring. En oversikt for 2012 utarbeidet av Umbrella Security Lab[42] bekrefter dette ved å gi et litt annet bilde av trussellandskapet. Her konkluderes det blant annet med at domener i TLD-er som *.ms*, *.com.co* og *.am* har en høy sannsynlighet for å være ondsinnet da over 75% av domenene i hver av disse TLD-ene skal være ondsinnet ifølge Umbrella Security Lab. Som det har blitt lagt frem i de nevnte rapportene produsert av kjente sikkerhetsfirmaer er risikoen større hos noen TLD-er. Noen TLD-er er ansett som risikofylte på grunn av deres høye antall ondsinnede domener(f. eks *.com*), og andre er ansett som risikofylte på grunn av deres høye andel ondsinnede domener(f. eks *.ms*). For å kunne definere terskel på hvilket TLD-er som er risikofylte må et helt oppdatert datagrunnlag benyttes. Dette vil være en dynamisk prosess hvor lister og verdier hele tiden må redefineres for å holde seg oppdatert med forandringene i trussellandskapet.

Land eller navn	**TLD**
Informasjon	INFO
Vietnam	VN
Cameroon	CM
Montserrat	MS
Armenia	AM
Colombia	COM.CO

Tabell 5: Forslag til TLD-er til bruk i prototypen

Utarbeidelsen av en slik oppdatert og gyldig oversikt etterlates til videre arbeid innen dette fagområdet. For praktiske årsaker har det blitt valgt å sette opp en enkel oversikt over TLD-er som burde observeres med datagrunnlaget som er nevnt her i rapporten for å gi en eksempelliste til bruk i prototypen som kommer i *Kapittel 5 - Prototype (s. 33)*. Den vil ikke ha tyngde nok til å kunne brukes i produksjon, men er heller en forenkling for å vise prototypen.

(10) Dato for opprettelse av domene

Som nevnt i *Avsnitt 3.1.2 - DGA (s. 11)*, hvis et botnet bruker DGA til å kalkulere navnet til C&C-serveren så må eieren av botnettet kun registrere et domenenavn for å kunne gi de infiserte maskinene nye instrukser. Da dette domenenavnet kun har som formål å fungere som en kommunikasjonskanal i et lite tidsrom opprettes disse domene ofte rett før kommunikasjonsprosessen trer i kraft. Holz et al.[7] påstår at alle domener generert av en DGA har et kort liv siden det kun skal brukes i en liten tidsperiode.

[7] brukte i sin utvikling av Exposure kort liv som et attribut som kunne skille godsinnede og ondsinnede domener. Tankegangen her var at godsinnede domener sjeldent plutselig ble opprettet og deretter fjernet etter en liten periode med mye trafikk. Denne tankegangen kan videreføres til at domener som har blitt opprettet innenfor en viss tidsperiode skiller seg ut fra normal oppførsel til godsinnede domener og kan være en kandidat til å bli klassifisert som et ondsinnet domene.

Det ble observert av Zhou et al.[43] at DGA-genererte domener skiller seg fra menneskegenererte domener ved blant annet livslengde. Yarochkin et al.[44] registrerte også meget lave verdier for hvor lenge domener levde i sine undersøkelser.

3.3 Konklusjon

I dette kapittelet har det blitt presentert et utvalgt av analyserbare attributer, en grunnmur for videre arbeid innenfor dette fagfeltet. De presenterte attributene er kun et eksempel på hva som kan brukes i et slikt system, og videre forskning vil med stor sannsynlighet identifisere enda flere attributer til å detektere skadevare. Bakmennene bak skadevaren vil utvikle bedre og smartere teknikker for å unngå deteksjon og nedtagelse av deres systemer, og sikkerhetseksperter må jobbe enda hardere med å finne nye måter å avdekke og stoppe disse teknikkene.

Sikkerhetseksperter har også en jobb å gjøre på eksisterende attributer. Skadevareteknikker kan utvikle seg ved å bygge på eksisterende teknikker, og deteksjon ved hjelp av de samme attributene kan være mulig hvis sikkerhetsmiljøet klarer å holde tersklene til disse verdiene ved like. Trussellandskapet er i konstant forandring og arbeidet med både eksisterende og nye attributer vil være en gjentagende prosess.

Deteksjon av skadevare er komplekst, og deteksjon ved kun å se på et enkelt attribut vil ofte ikke være nok. Tanken er at attributer skal kunne kombineres for å finne stadig mer komplekse teknikker, både attributer nevnt i denne rapporten og attributer avdekket ved videre forskning. Tidligere i rapporten ble signaturbaserte deteksjonssystemer og statistisk anomalibaserte deteksjonssystemer diskutert da disse er de mest fremtragende i dagens systemer. Ved utvikling av tester av attributer må det tas et valg om hvilken fremgangsmåte som skal brukes.

4 Kravspesifikasjon og design

4.1 Kravspesifikasjon

Det er ønsket fra oppdragsgiver å identifisere kravene og behovene til et rammeverk som denne oppgaven skal fokusere på - et rammeverk for analyse av pDNS-data. Dette kapittelet er utarbeidet i tett samarbeid med oppdragsgiver for å best kunne levere et produkt som gjenspeiler deres ønsker og krav. Rammeverket som skal utarbeides er tiltenkt å kjøres primært av en analytiker i sitt arbeid med å analysere sikkerhetshendelser, men kan også benyttes som en automatisk prosess f.eks *daemon*-tjeneste eller *cron-job*. Sikkerhetslandskapet er i konstant forandring, og trusselaktørene er stadig på jakt etter nye metoder for å skjule sine intensjoner og aktiviteter. Med dette som grunnlag ble det ønsket å utvikle et system som raskt lar seg videreutvikles og utvides ved behov, og som skal benytte en plugin-basert struktur for hva slags alarmer den skal produsere og hvilke analyserbare attributter det skal sees etter. Programmet skal primært motta input gitt fra en bruker, enten dette er stien til en fil som inneholder pDNS-data, eller medsendte kommandolinjeargumenter om en spesifikk DNS-query.

mnemonic har utplassert et større antall pDNS-sensorer på forskjellige lokasjoner over store deler av verden. Denne internasjonale spredningen gjøres mulig både via mnemonic sine egne kunder, men også andre samarbeidspartnere som faller utenfor deres kundelister. Disse sensorene er utplassert på nettverkspunkter og lytter til DNS-trafikk på som flyter forbi sine respektive knutepunkter. Mer spesifikt så fanger disse sensorene inn både vellykkede og ikke vellykkede queries ettersom hvor de er plassert. Disse innsamlede dataene sendes tilbake til en sentralisert og aggreggert pDNS-DB hos mnemonic. Rammeverket er logisk plassert ved mnemonic sin aggregerte pDNS-DB som vist i *Figur 4 - Skisse over plassering av rammeverket (s.19)*. Denne logiske plasseringen gjenspeiler ikke den fysiske, og en analytiker kan sitte på vidt forskjellige lokasjoner. En analytiker kan også ha sitt eget datasett eller individuelle domenenavn de vil analysere. Tanken er at den logiske plasseringen gjenspeiler at det ikke er noe annet system, programvare eller aktivitet som må foregå før rammeverket kan analysere, selv om dette er mulig såfremt datasettet er på gyldig format.

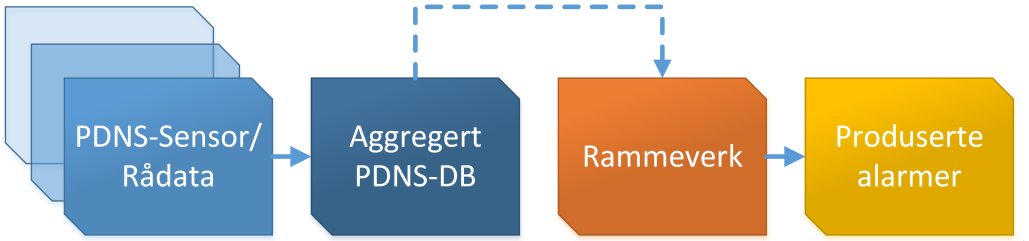
4.1.1 Mål

Resultatmål

- Det skal identifiseres behovene til et rammeverk for analyse av et pDNS-datasett som presenteres i form av et kravspesifikasjon- og designdokument.
- Det skal utvikles en prototype av rammeverket som oppfyller alle funksjonelle krav gitt i kravspesifikasjonsdokumentet.
- Den utviklede prototypen skal inneholde en eller flere plugins som tilbyr analyse og deteksjon av en potensiell indikator for ondsinnet aktivitet.

Effektmål

- Rammeverket skal kunne tilbys til sikkerhetsanalytikere som et verktøy for å raskere kunne analysere pDNS-data enn dagens løsninger kan tilby.
- Rammeverket skal kunne bli benyttet i andre løsninger og programmer som har behov for en fleksibel og tilpassningsdyktig analyse av pDNS-data.
- De identifiserte kravene til et slik rammeverk skal kunne fremskynde en potensiell utviklingsprosess hos oppdragsgiver.



Figur 4: Skisse over plassering av rammeverket

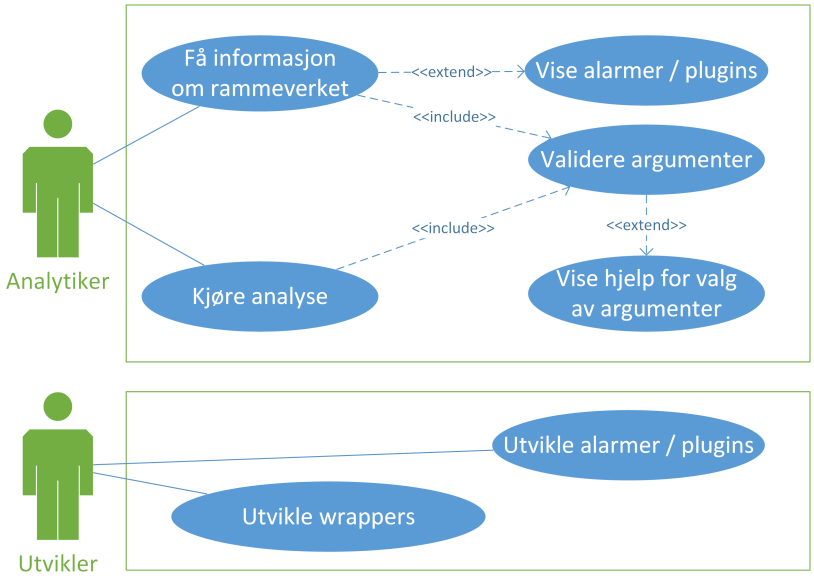
4.1.2 Funksjonelle krav

Listen under er en overordnet forklaring på hvilken funksjonalitet rammeverket som et ferdig produkt skal kunne tilby både en sluttbruker, og en utvikler som potensielt skal vedlikeholde eller videreutvikle programvaren.
Rammeverket skal kunne:

- Lese inn en eller flere pDNS-oppføringer om gangen.
- Lese inn et pDNS-datasett fra fil.
- Lese inn en pDNS-oppføring gitt via kommandolinjeargumenter.
- Gi output på både rådata og stilsatt format.
- Gi et resultat / output-linje per plugin som hver pDNS-oppføring analyseres i.
- Levere output som skrives til skjerm.
- Levere output som skrives til en fil.
- Gi brukeren mulighet til *verbose* utskrift av handlinger foretatt i rammeverket (for debugging).
- La brukeren administrere hvilke analyseplugins som skal benyttes under en gitt kjøring av rammeverket.
- La brukeren se informasjon om rammeverkets potensielle kommandolinjeargumenter.
- La brukeren se informasjon og beskrivelse om de forskjellige pluginene som er lagt inn i rammeverket.
- La en utvikler benytte rammeverket som en modul i et eget programvare (for eksempel via et *wrapper*-skript eller lignende).
- La brukeren sette parametere for kjøring via kommandolinjeargumenter.

4.1.3 Interaksjon med rammeverket

Det er ønsket fra oppdragsgiver at dette rammeverket er et verktøy som kan kjøres av forskjellige typer analytikere for å videre kunne undersøke potensielle sikkerhetshendelser. Det ble også ytret et ønske om å kunne kjøre programmet som en daemon-tjeneste. Dette er noe som enkelt vil løses ved å ha en struktur og oppbygning av rammeverket som enten tillater alternative metoder for kjøring, som å inkludere rammeverket i et enkapsulerende program for kjøring som daemon eller ved å legge til rette for direkte videreutvikling. For denne løsningen har en kombinasjon av de to metodene blitt valgt. Programmet skal utvikles som et program som *ut-av-boksen* kan kjøres på forskjellige måter. Den skal inkludere en løsning for å kunne kjøres direkte som et verktøy fra et Command-line interface (CLI). Denne løsningen skal tilby et fullverdig grensesnitt i form av feilhåndtering av feilaktig brukerinput og hjelpefunksjoner til bedre kunne forstå bruken av rammeverket. Samtidig skal det også være mulig å benytte programmet via wrappere eller importering (som f.eks `import palmadns` i python eller `#include <palmadns.h>` i C++). Et eksempel på en mulighet for å kjøre rammeverket via andre programmer eller løsninger er å skrive et wrapper program og en mulighet her er å skrive et bash-skript som en wrapper. Et eksempel på dette finner man i *Appendix C.5 - /large_file_analysis.sh (s. 141)*.



Figur 5: UML use case-diagram (høynivå)

Videre i dette avsnittet vil betegnelsene analytiker, bruker og sluttbruker benyttes om hverandre. Med alle disse tilfellene så menes det en bruker av rammeverket som ikke skal utføre vedlikehold, videreutvikling eller lignende. For alle slike aktiviteter vil disse betegnes som administratorer. Et høynivå use case-diagram vises i *Figur 5 - UML use case-diagram (høynivå) (s.20)*. Her er det trukket ut kjernen av hva rammeverket skal tilby en analytiker. Videre detaljer om de forskjellige elementene i diagrammet som f.eks validering av argumenter er forklart i dybde i dette kapittelet.

Kommandolinjeargumenter

For at rammeverket skal kunne levere mer enn bare statisk funksjonalitet til brukeren er det nødvendig å kunne variere ønsket handling og eller begrensninger på ønskede

handlinger. *Tabell 6 - Forklaring av kommandolinjeargumenter (s. 21)* viser en oversikt over alle kommandolinjeargumenter det er mulig å gi til rammeverket når det kjøres som et skript i et CLI-miljø. Det er også vist forskjellige grupperinger av argumenter som kan gis. Disse kan sees på som undermenyer i rammeverket. Dette gjøres ettersom det er forskjellige muligheter og krav til input fra brukere hvis det er en fil som skal analyseres, eller det er dokumentasjon om rammeverket brukeren vil ha skrevet til skjerm. Felles for begge analysehandlingene er at man kan velge å spesifisere hvilken plugins man skal kjøre via `-plugins`, og om resultatene skal skrives til en fil via `-o / -output`. Et aktivitetsdiagram over denne flyten vises i *Figur 34 - UML-aktivitetsdiagram for validering av kommandolinjeargumenter (s.61)*.

Kommandolinjeargumenter			
Handling	Operatør	Beskrivelse	Valgfri
Kategorier for kjøring	run	Skrives ved starten av alle argumenter for å indikere at det er en analyse som skal kjøres. Dette argumentet åpner for videre bruk av `run file` eller `run params`.	N/A
	run file	Skrives ved starten av alle argumenter for å indikere at det er en analyse som skal kjøres, og denne skal analysere en fil. Dette gjøres for å angi hvilke påfølgende parametere som er gyldige.	N/A
	run params	Samme som ved `run file` bare at den analyserer parametere gitt som argumenter.	N/A
	information	For å etterspørre mer informasjon.	N/A
Felles for analyse av datasett fra fil og fra argumenter	–formated	Angir om utskrivning til skjerm skal vises som rådata eller formatert og finere.	Ja
	-o / –output	Filen som programmet skal skrive alarmresultatene til.	Ja
	-v / –verbose	Gir sluttbrukeren mulighet til å vise mer informasjon ved kjøring av programmet.	Ja
	-h	Viser en hjelpemeny for hvilke mulige argumenter man kan sette.	Ja
	-p / –plugins n...	Hvilken plugins / alarmer man vil aktivere. Kun disse vil kjøres.	Ja
Datasett fra fil	-i / –input	Filen som programmet skal lese inn pDNS-datasettet fra.	Nei
Datasett fra argumenter	-q / –query	Angir ønsket domenenavn som skal analyseres.	Nei
	–answer	Angir hvilken IP angitte domene resolver eller har resolvet til.	Ja
	–ttl	Angir hvilken TTL som ble observert ved siste query.	Ja
Informasjon	–about	Informasjon om programmet.	Ja
	–license	Lisensinformasjon om programmet.	Ja
	–list	Viser alle tilgjengelige plugins.	Ja

Tabell 6: Forklaring av kommandolinjeargumenter

I *Appendix A.1 - Kjøring av rammeverket som fil (s. 60)* er det fremstilt tre mulige

metoder for kjøring av rammeverket som et selvstendig program. Linje 1 viser analyse av filen `pdns.csv` hvor output skal gå til `log.csv` samt at det kun er alarmen `PLUGIN-001` som skal kjøres. Linje 2 viser analyse av parametere ved kjøring, her er det domenet *www.example.org* som skal analyseres med TTL-verdi er satt til 60. Til slutt viser linje 3 henting av informasjon om alle plugins i rammeverket.

```
[1] >> python PalmaDNS run file --input pdns.csv --output log.csv -p PLUGIN-001
[2] >> python PalmaDNS run params -q www.example.org --ttl 60
[3] >> python PalmaDNS information --list
```

Figur 6: Eksempler på kjøring av rammeverket via kommandolinje

Brukergrensesnitt

Det er ikke satt noe krav eller ønske om at dette rammeverket skal kunne benyttes ved bruk av et grafisk brukergrensesnitt, men det er derimot ønsket at programmet er utviklet for bruk via et CLI-grensesnitt. Ettersom målgruppen for dette rammeverket er primært analytikere med særskilt IKT-kompetanse vil ikke dette valget medføre særlige vansker for sluttbrukerene. Rammeverkets struktur og modularitet vil derimot legge til rette for videreutvikling slik at andre grensesnitt, som for eksempel et web-grensesnitt, kan benyttes.

4.1.4 Datakilder

De følgende avsnittene tar for seg de forskjellige input- og output-formatene som rammeverket skal basere seg rundt. Først og fremst er det CSV-formater som er benyttet når det gjelder input og output. Disse formatene vil bli benyttet når det gjelder lesing og skriving til fil ettersom dette primært sett skal være maskinleselige formater. Samtidig skal det finnes et stilsatt format for output for å skape et format som er mer leselig for mennesker.

CSV-format - Input

Fra oppdragsgiver fikk prosjektgruppen overlevert et utdrag av en pDNS-logfil. *Figur 7 - CSV-format for pDNS-loggfil (s.23)* viser et eksempel på dette. Den følger en CSV-stil hvor *delimiteren* er `||` og filen har en *header* samt 8 felter med data om hver query. Dette dataformatet er brukt ettersom pDNS-sensoren som ble benyttet til produksjon av denne loggen er programmet *PassiveDNS* laget av Edward Bjarte Fjellskål[1], og dette programmet bruker `||` som delimiter. Det finnes som nevnt tidligere mange forskjellige implementasjoner av pDNS-sensorer, og hittil er det ikke bestemt noe standard output-format. Det er pågående arbeid med en Internet Draft for en slik standardisering[2] drevet av representanter fra den Østeriske CERTen. Loggfilene som skal kunne importeres inneholder de følgende datafeltene.

Timestamp Timestamp-verdi (epoch/Unix timestamp).

DNS client Klienten som utførte DNS-queryen.

DNS server DNS-serveren som mottok queryen og behandlet den.

RR Class Resource Records-klassen.

[1] https://github.com/gamelinux/passivedns

[2] `https://datatracker.ietf.org/doc/draft-dulaunoy-kaplan-passive-dns-cof`

Query Domenet som det ble forespurt om i queryen.

Query Type Typen svar som ble gitt som resolve.

Answer IPen eller Peker til.

TTL TTL-verdi for dette domenet ved queryen.

```
Timestamp||DNS client||DNS server||RR Class||Query||Query Type||Answer||TTL
1389322791||192.168.0.31||192.168.1.1||IN||www.example.com||A||10.0.0.12||3600
1389322792||192.168.0.22||192.168.1.1||IN||www.example.org||A||10.0.0.11||60
1389322795||192.168.0.31||192.168.1.1||IN||www.example.com||A||10.0.0.22||800
1389322801||192.168.0.12||192.168.1.1||IN||www.example.net||A||10.0.0.10||60
```

Figur 7: CSV-format for pDNS-loggfil

Dette formatet legger grunnlaget for hva slags informasjon og eventuelt analyserbare attributer rammeverket vil ha tilgang til. Rammeverket skal i utgangspunktet utvikles for å gjenspeile det datasettet som er tilgjengelig og tiltenkt brukt i denne omgang. Hvis man møter på situasjoner hvor man har data fra andre typer sensorer eller har eksportert data fra andre sikkerhetsprodukter vil rammeverket ikke kunne håndtere dette. Dette kan potensielt være begrensende for hva slags situasjoner eller potensielt truende hendelser dette rammeverket kan detektere. Det har ikke blitt foretatt noen vurdering om dette datasettet inneholder tilstrekkelig med data med tanke på hva man kan detektere ved slike type pDNS-sensorer. Prosjektet tar utgangspunkt i at dette er det tilgjengelige datasettet og må bygge rammeverkets krav og funksjonalitet ut i fra dette.

Denne situasjonen med et potensielt begrenset datasett kan løses på to mulige måter. Det er mulig å utvikle individuelle skripts til å konvertere dataformatene til noe som rammeverket kan forstå. Med tiden vil dette føre til et større antall små skripts som kan skape en uoversiktlig situasjon. Det andre alternativet er å omstrukturere rammeverket i en fremtidig utgave som lar brukeren definere - for eksempelvis i en konfigurasjonsfil - de forskjellige dataformatene. Deretter kan hvilket dataformat som er brukt enten detekteres av rammeverket eller brukeren kan velge dette eksplisitt ved kjøring av rammeverket. Dette vil føre til større arkitekturmessige endringer, men kan bringe med seg mange fordeler hvis et slikt valg tas i fremtiden.

Outputformat - rådata

CSV-filer har ingen formell godkjent standard, og med dette finnes det mange friheter med tanke på formatet man bruker i sin CSV-fil. Et eksempel på dette er som beskrevet i *Figur 7 - CSV-format for pDNS-loggfil (s.23)* med bruk av `||` som delimeter. Det har blitt valgt å følge den defacto-standarden[45]. Rammeverket skal som nevnt produsere en ouput per alarm som er analysert på hver DNS-query. Disse alarmene skal inneholde domenenavnet, IP-en denne resolver til, analysen som har blitt kjørt samt en resulterende *score* til denne alarmanalysen. Hvilken score som blir gitt kommer ann på hva den respektive pluginens vurdering av ondsinnhet for den gjeldende oppføringen. Mer informasjon om dette konseptet er beskrevet i *Avsnitt 4.2.3 - Plugins (s. 29)*.

Som man ser i de kommende beskrivelsene av feltene og *Figur 8 - CSV-format for resulterende analyse (s.24)* er det kun et utvalg av data som skrives ut i hver alarm. Det kommer mer data inn til rammeverket i form av de forskjellige attributene som vist i *Figur 7 - CSV-format for pDNS-loggfil (s.23)*. Dette er noe som potensielt kan hindre

andre systemer som skal videre analysere rammeverkets output. Rammeverkets kode burde da produseres slik at dette er lett å endre. Potensielle løsninger her er å utvikle og tilpasse output-behov når det kreves. En annen løsning er å gjøre noe lignende som forslaget i *Avsnitt 4.1.4 - CSV-format - Input (s. 22)* med tanke på varierende input-formater ved at man kan la brukere skrive konfigurasjonsfiler for forskjellige output-formater. De forskjellige feltene som er med i dette formatet er:

Client IP-en til den klienten som har sendt denne DNS-queryen.

Answer Svaret som ble returnert tilbake som ønsket domene resolver til.

Query Domenenavnet som det ble sendt en query om.

Plugin Pluginen som har blitt analysert og som denne oppføringen gjelder for.

Score Score / resultatet som denne alarmen har gitt følgende query med tanke på sin respektive analysefunksjon.

```
Client,Answer,Query,Plugin,Score
192.168.0.31,10.0.0.12,www.example.com.,001-TLD,0.5
192.168.0.22,10.0.0.11,www.example.org.,002-TTL,0.1
192.168.0.31,10.0.0.22,www.example.com.,001-TLD,0.5
192.168.0.12,10.0.0.10,www.example.net.,002-TTL,0.1
```

Figur 8: CSV-format for resulterende analyse

Følgende er regler er satt for formatet i output CSV-formatet:

1. Første linje av filen vil være en header til å forklare de individuelle feltene.
2. Ingen av feltene vil inneholde noen anførselstegn (") eller være enkapsulert av dem.
3. Delimiteren mellom feltene vil være komma (,), ettersom ingen av feltene vil eller skal inneholde dette tegnet. Det ble besluttet å heller følge den defacto standard[45] for CSV-filer enn formatet brukt i pDNS-loggfilene som vist i *Figur 7 - CSV-format for pDNS-loggfil (s.23)*. Oppdragsgiver har ingen avhengighet eller preferanser mot det ene eller andre formatet, og valget falt dermed på den mest universale løsningen.

Outputformat - stilsatt

Når en analytiker skal gjennomføre analyse av et mindre antall oppføringer i et utdrag av en loggfil eller velger å analysere innskrevne argumenter, kan det være ønskelig å få resultatene rett til skjerm. Ved større filer vil det ikke være hensiktsmessig å få flere hundretusen analyseresultater fra et titalls plugins skrevet til skjerm. Når en bruker ikke har indikert noen destinasjon for output-fil (ved `-output / -o` argumentet) vil rammeverket skrive resultatet til skjermen. For å bedre kunne tolke og gi en bedre brukeropplevelse er det mulig å indikere et ønske om en formatert utskrift ved argumentet `-formated`. *Figur 9 - Format for visning av resulterende analyse (s.25)*. Hvis argumentet `- formated` ikke er gitt ved kjøring av programmet vil programmet skrive ut på samme formatet som ved skriving til fil som vist i *Figur 8 - CSV-format for resulterende analyse (s.24)*.

```
Client              | Answer              | Query               | Alarm     | Score
-------------------------------------------------------------------------------------
192.168.0.31        | 10.10.0.12          | www.example.com.    | 001-TLD   | 0.56
192.168.0.22        | 10.200.0.11         | www.example.org.    | 002-TTL   | 0.01
192.168.0.31        | 10.0.0.222          | www.example.com.    | 001-TLD   | 0.75
192.168.0.12        | 10.0.0.10           | www.example.net.    | 002-TTL   | 0.12
```

Figur 9: Format for visning av resulterende analyse

Output som returneres til kallende funksjon

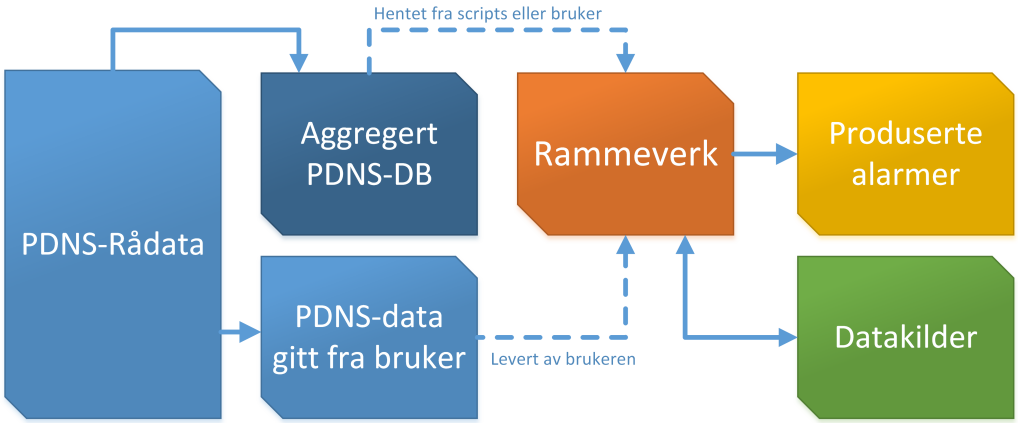
Ettersom rammeverket skal kunne benyttes i andre programmer enten som en wrapper eller inkludert i ny kode - som eksempelvis `import palmadns` hvis utviklet i Python eller `#include <palmadns.h>` hvis utviklet i C++ - skal det også være mulig å få resultatene returnert. Dataformatet for dette er enkelt å forandre i koden. Gode valg kan være å levere dette som en liste eller lignende datastrukturer som gjøre det enklere å behandle hvert resultat individuelt kontra en lengre streng som krever nok en runde med behandling som igjen vil kreve mer ressurser. Når man velger at output-destinasjon skal være å bli returnert tilbake til den kallende funksjonen vil man kunne velge om det skal være formatert eller rådata. Mest praktisk her vil være rådata-formatet, men mulighetene burde være veldig åpne og frie for andre utviklere, og derfor burde også det stilsatte formatet være mulig å velge her. Et eksempel på denne formen for bruk av rammeverket vises i *Figur 10 - Eksempel på løsning hvor rammeverket returnerer resultater (s.25)*.

```
1  #!/usr/bin/python
2
3  def run():
4      # Create an instance of the core
5      core = core.Core()
6      core.set_output_destination(core.DESTINATION_RETRUN)
7      core.set_output_format(core.FORMAT_RAW)
8      core.import_file('pdns.log')
9
10     # Run an analysis and print out the results
11     results = core.run()
12     for result in results:
13         print result
14
15 if __name__ == '__main__':
16     # Try importing the PalmaDNS package
17     try:
18         import PalmaDNS.core
19     except ImportError, e:
20         print '[ ERROR ] - ', e
21     else:
22         run()
```

Figur 10: Eksempel på løsning hvor rammeverket returnerer resultater

4.2 Organisering og arkitektur

Som beskrevet i løpet av *Avsnitt 4.1 - Kravspesifikasjon (s. 18)* vil rammeverket være logisk plassert slik at det skal analysere nye pDNS-oppføringer. Dette kan gjøres enten ved å sette opp forskjellige automatiske analyseskripts som kjører rammeverket periodisk, eller ved andre løsninger som ikke krever brukerinteraksjon. Primært så skal rammeverket være en brukerinteraksjonsstyrt enhet. Når en analytiker ser mistenklig aktivitet fra andre sensorer på et nettverk kan de hente ut pDNS-data fra pDNS-DBen og dermed ha mer informasjon tilgjengelig for å kunne ta en mer begrunnet vurdering i løpet av en *triage*-prosess. Selv om rammeverket ikke ser noe på koblingen opp mot pDNS-DB-en kan det være nyttig å vurdere utvikling eller inkludering av funksjonalitet som kan hente ut ønskede oppføringer knyttet til en gitt hendelse. Selv om rammeverket er logisk organisert slik at den er koblet mot en pDNS-DB vil ikke dette rammeverket fokusere noe på en eventuell kobling opp mot denne databasen. *Figur 11 - Skisse over plassering av rammeverket (s.26)* viser denne plasseringen i mer detalj og de omkringliggende delene knyttet til rammeverket. *Datakilder*-elementet i denne figuren viser til alt av informasjon som rammeverket kommer til å benytte som ikke er innkommende pDNS-oppføringer. Dette kan være alt i fra statisk data som konfigurasjonsfiler til dynamisk data som f.eks spørringer om WHOIS informasjon eller geolokasjonsoppslag.



Figur 11: Skisse over plassering av rammeverket

For å imøtekomme et ønske om en fleksibel og utvidbar struktur samt en alarmproduksjon som oppfyller de krav oppdragsgiver har stilt falt valget på å utvikle et modulært system med et rammeverk i midten til å knytte alt sammen. Moduler av dette systemet er arbeidsoppgaver som varierer fra å holde orden på en liste over pDNS-oppføringer, håndtering av plugins, validering av kommandolinjeargumenter til en kjerne for å knytte alt dette sammen. En skisse av denne strukturen presenteres i *Figur 12 - Skisse av koblingene mellom rammeverkets moduler (s.27)*. Rammeverket står som ansvarlig for mottak av input fra de potensielle pDNS-kildene (loggfil eller brukerinnskrevet) samt ouput av de produserte alarmene til ønsket format og destinasjon (skjermvisning, fil, eller som `return` til en kallende funksjon). Brukeren og eller administratoren av systemet kan definerer så mange alarmer de har behov for og disse vil legges inn i rammeverket automatisk hvis all nødvendig data er konfigurert riktig. Samtidig er det også mulig å spesifisere ved kjøringen av rammeverket hvilke plugins man vil ha aktivert. Hver alarmplugin er en

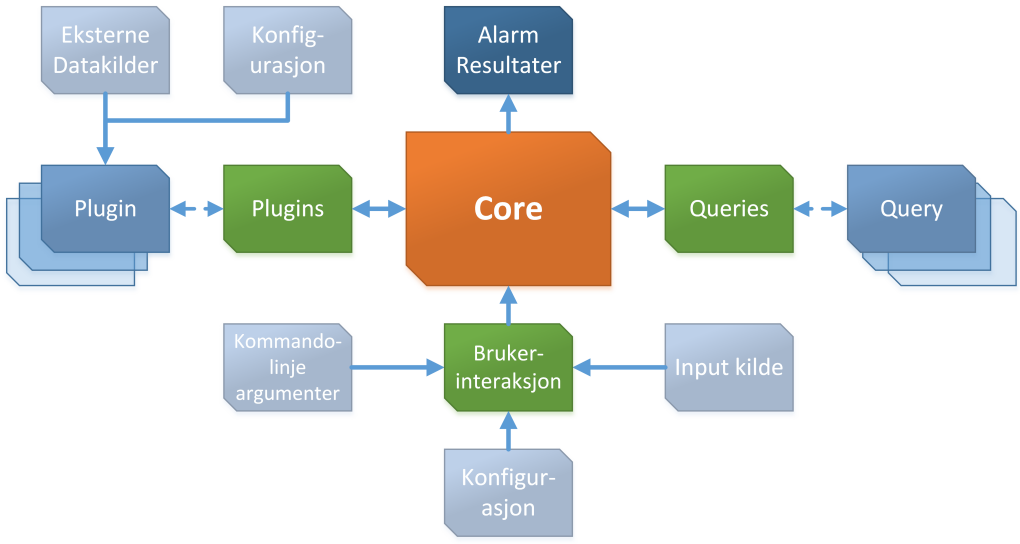
selvstendig analyseenhet som eksisterer i sin egen fil. Disse kan enkelt slettes, redigeres eller utvides ved behov.

Hendelsesflyt

For å bedre kunne forklare og formidle den konkrete flyten til rammeverket er det utarbeidet flere Unified Modelling Language (UML)-aktivitetsdiagrammer som ligger i *Appendix A - Figurer og diagrammer (s. 60)*. Det første aktivitetsdiagrammet *Appendix A.1 - Kjøring av rammeverket som fil (s. 60)* skildrer hendelsesflyten når brukeren av systemet velger å kjøre programmet som en selvstendig fil. Dette vil forøvrig være den mest naturlige og hyppige metoden for bruken av rammeverket, selv om andre metoder er mulige. Hovedfokuset her er å lese inn brukerens medsendte argumenter slik beskrevet i *Avsnitt 4.1.3 - Kommandolinjeargumenter (s. 20)*, og deretter sende brukerne til riktig funksjonalitet i rammeverket. Dette diagrammet har flere forklarende notiser for å gjøre flyten mer forståelig. Et valg som ble tatt er å ikke foreta dypdykk videre inn i underliggende moduler (som f.eks kjøring av analyse i `Framework`-modulen). Videre så er *Appendix A.3 - Gjennomføring av analyse (s. 62)* et dypdykk inn i hvordan en analyse vil foregå i rammeverket. Her følges det samme prinsippet om å ikke foreta dypdykk inn i noen andre moduler (f.eks ved aktivering av plugins) enn den som skildres i diagrammet.

4.2.1 Moduler

Som tidligere forklart skal dette rammeverket være organisert slik at de forskjellige hovedfunksjonalitetene skilles ut i egne moduler samt at rammeverket skal ha en hovedmodul som knytter alle disse modulene sammen. En skisse over denne arkitekturen vises i *Figur 12 - Skisse av koblingene mellom rammeverkets moduler (s.27)*. Systemet vil videre bli beskrevet via de forskjellige sentrale modulene / funksjonalitetene: `Brukerinteraksjon`, `Core`, `Plugins` og `Queries`. Grunnlaget for en slik fordeling er todelt. Ved å gjøre det slikt beholdes en objektorientert tankegang med utviklingen av rammeverket samtidig som de invididuelle modulene og komponentene eventuelt kan gjenbrukes i andre systemer eller løsninger.



Figur 12: Skisse av koblingene mellom rammeverkets moduler

Brukerinteraksjon

Ettersom programmet skal kunne kjøres av analytikere kreves det en samling av funksjonaliteter som håndterer denne interaksjonen. Modulen `Brukerinteraksjon` skal være til for nettopp dette. Ettersom rammeverkets hovedfunksjonalitet (som beskrevet i `Framework` modulen) skal kunne kjøres selvstendig, men også som initiert av en analytiker, ble det fornuftig å skille all brukerinteraksjon ut av denne modulen og inn i en separat modul for å håndtere situasjonene hvor rammeverket kjøres som et selvstendig program. Modulens primære funksjon er at det er den som kalles / kjøres fra kommandolinjen når brukeren ønsker å kjøre rammeverket. I tillegg til dette så oppretter modulen de nødvendige objektene av rammeverket som kreves for å kjøre. Videre så skal denne modulen håndtere og validere kommandolinjeargumenter for å kunne henvise brukeren til de riktige funksjonene som ble etterspurt.

Denne modulen har tre input-kilder. Først er det som beskrevet kommandolinjeargumentene. Herfra kan brukerne sette ønsket handling fra rammeverket, input- og eller output-kilde og eventuelt etterspørre mer informasjon om rammeverket. Videre vil denne modulen lese inn eventuelle konfigurasjonsfiler. Dette kan være alt fra konfigurasjon for loggføring eller ferdige konfigurerte output-destinasjoner hvis slikt er ønskelig å videreutvikle. Til slutt har også denne delen ansvar for videresending av parametere som input-fil eller input-parametere slik at `Framework`-modulen kan analysere det brukeren har ønsket skal bli analysert.

Core

`Core` skal sees på som hovedmodulen i hele rammeverket. Den fungerer som et lim mellom all annen funksjonalitet som er tiltenkt i programmet, samtidig som det er den som står ansvarlig for initiering av analysen samt utskrivning av resultatene til ønsket destinasjon. Etter at modulen for brukerinteraksjon er ferdig med sitt arbeid vil den kalle ønsket funksjonalitet som grovt sett kan deles inn i visning av mer informasjon om rammeverket, og kjøring av analyse. Denne modulen oppretter et objekt av modulen `Plugins` og et av modulen `Queries`. Ved å gjøre dette har da modulen tilgang på et enkelt grensesnitt for kjøring av analyseplugins, samt et enkelt grensesnitt for organisering og tilgang til de innleste pDNS-oppføringene. Denne modulen er også ansvarlig for utskrift av resultatene fra hver enkelt analyse. Det er gitt fra brukerinteraksjonsmodulen hvor disse resultatene skal ende opp (skjerm eller fil) samt hvilket format de skal ha. Denne modulen vil da ta hånd om denne utskrivningen.

Plugins

`Plugins`-modulen i rammeverket skal stå ansvarlig for håndtering, aktivering/deaktivering og tilgjengeliggjøring av de forskjellige analysefunksjonene en utvikler har lyst til å produsere, eller som en analytiker vil benytte seg av. Hver slik analysefunksjon er representert via en selvstendig plugin i en selvstendig fil og det er dermed ønskelig med et system som kan håndtere dette på en oversiktlig og fleksibel måte. En mer omfattende forklaring av funksjonaliteten og arkitekturen til denne modulen finnes i *Avsnitt 4.2.3 - Plugins (s. 29)*. Overordnet sett så skal denne modulen være et grensesnitt for `Core`-modulen for å kunne aktivere og eller deaktivere ønskede plugins, vise sluttbrukeren en total oversikt over alle plugins, samt gi tilgang til analysefunksjonene i hver plugin når `Core`-modulen skal gjennomføre sin analyse av ønsket pDNS-datasett.

Queries

`Queries`-modulen skal stå ansvarlig for innlesning og håndtering av alle pDNS-oppføringer som blir gitt til rammeverket. Målet med denne modulen er å skape et minimalistisk grensesnitt for håndtering av en potensiell svært stor mengde med pDNS-oppføringer. `Queries`-modulen vil bygge opp en liste (eller lignende datastruktur) over alle innleste oppføringer som representeres i hvert sitt `Query`-objekt. En mer omfattende beskrivelse kan finnes i *Avsnitt 4.2.4 - Queries (s. 31)*. `Queries`-modulen vil være tilgjengelig som et objekt i `Core`-modulen. `Core`-modulen vil sende inn sitt datasett (uansett om dette er et filnavn eller kommandolineargumenter) og i ettertid ha tilgang på en liste over `Query`-objekter som representerer en enkelt oppføring.

4.2.2 Filstruktur

For å bedre fremstille og vise de forskjellige modulene er det utarbeidet et forslag til en filstruktur for rammeverket. Eksempelet under er en filstruktur som tar utgangspunkt i et system laget i programmeringsspråket Python. Videre så er det organisert i en pakke-struktur hvor det skal kunne gå ann å benytte mappestrukturen som et individuelt skript (som vist i *Figur 6 - Eksempler på kjøring av rammeverket via kommandolinje (s.22)*) eller inkludere det i andre pythonprogrammer (som vist i *Figur 10 - Eksempel på løsning hvor rammeverket returnerer resultater (s.25)*). Dette forslaget presenteres i *Figur 13 - Forslag til filstruktur (s.29)*. Hvis denne figuren settes opp mot moduloversikten i *Figur 12 - Skisse av koblingene mellom rammeverkets moduler (s.27)* vil det kunne forklares slik:

```
\
|--- __main__.py
|--- arguments.py
|--- \PalmaDNS
|----|--- core.py
|----|--- plugins.py
|----|--- queries.py
|----|--- query.py
|----|--- \plugins
|---------|--- Plugin1.py
|---------|--- Plugin1.cfg
|---------|--- [...]
|----|--- \config
|---------|--- config.ini
|---------|--- [...]
```

- Brukerinteraksjon er i `__main__.py`
- Kommandolinjeargumenter er i `arguments.py`
- Konfigurasjon er i `PalmaDNS/config`
- Framework er i `PalmaDNS/core.py`
- Plugins er i `PalmaDNS/plugins.py`
- Hver plugin: `PalmaDNS/plugins/*.py`
- Hver plugin-konfigurasjon: `PalmaDNS/plugins/*.cfg`
- Queries er i `PalmaDNS/queries.py`
- Hvert query-objekt er i `PalmaDNS/query.py`

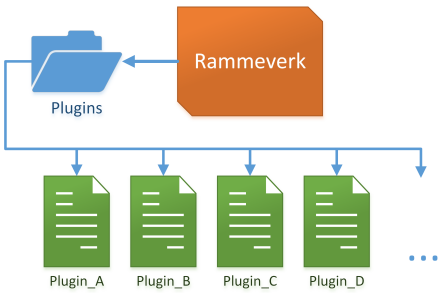
Figur 13: Forslag til filstruktur

4.2.3 Plugins

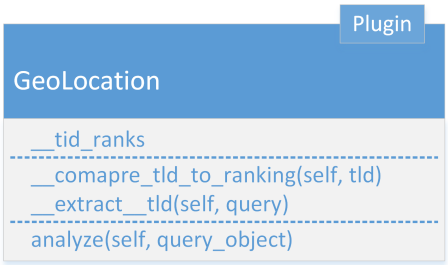
Når man snakker om en plugin med tanke på rammeverket så menes det en utvidelse av systemet designet for å motta et standardisert sett av data, gjennomføre en analyse etter det den respektive modulen ser etter av ondsinnede indikatorer og returnere en standardisert verdi som indikerer graden av mistenklighet for at den respektive DNS-querien var ondsinnet.

Som tidligere beskrevet i dette kapittelet har det blitt ønsket et fleksibelt system og struktur for håndtering av plugins til rammeverket. Metoden som ble valgt da var at hver plugin eksisterer som en individuell fil med en gitt mal og struktur samt konfigurasjonsfil

for å kunne gi brukeren mer informasjon om den gitte pluginen. *Figur 14 - Hierarkiet til plugins (s.30)* viser hvordan rammeverket vil ha en faktisk mappe som inneholder alle plugins som skal brukes til analyse. Det finnes mange metoder for å implementere et slikt plugin-system. Man kan lage noe eget selv, men det finnes allerede mange ferdiglagde og velfungerende systemer for dette. Et eksempel som rapporten kommer tilbake til i *Kapittel 5 - Prototype (s. 33)* er bruken av Yet Another Plugin SYstem (YapSY) til Python. Denne organiseres på måten beskrevet over med individuelle filer og konfigurasjonsfiler per plugin og tilbyr et svært fleksibelt system for håndtering av plugins.



Figur 14: Hierarkiet til plugins



Figur 15: Klassediagram for eksempel-plugin

Antallet produserte alarmer

Når en analyse kjøres vil det være alt i fra en til hundrevis, kanskje milioner av pDNS-oppføringer som kjøres igjennom rammeverkets analysefunksjonalitet. For å best kunne skaffe en oversikt over alle resultater ble det ønsket at hver query skulle analyseres i hver plugin. I tillegg til dette skal hver plugin sitt resultat skrives ut til ønsket destinasjon. En matematisk representasjon av dette forholdet mellom antall plugins og antall DNS-queries vises i *Figur 16 - Antall produserte alarmer (s.30)*. Enkelt forklart så innebærer denne metoden en N-dobling av antall linjer med resultater for N antall plugins. Eksempelvis, hvis en analytiker velger å analysere en loggfil med 200 queries og rammeverket inneholder 10 plugins, så vil den resulterende utskriften være på 2000 linjer. Fra starten av prosjektet ble det i samarbeid med oppdragsgiver sett på muligheten for å korrelere de forskjellige resultatene og aggregere samt vekte disse opp mot hverandre for bedre kunne si noe om hver enkelt pDNS-oppføring. Fordeler ved dette hadde vært at man fikk et mindre antall resultater produsert per oppføring. Samtidig så ville en slik løsning innebære at man raskere fikk ut et svar på om analysen indikerer noe potensielt ondsinnet. I et scenario hvor man har utviklet 50 plugins og det analyseres en pDNS-oppføring vil man måtte se igjennom resultatene på de 50 forskjellige plugins-outputene samt vurdere helheten. Målet med dette rammeverket er ikke å gjennomføre en slik korrelering og vekting av de forskjellige pluginene. Dette ble vurdert som en for kompleks og avansert oppgave for denne rapporten å ta for seg da prosjektgruppen ikke har datagrunnlaget som trengs for å korrelere resultater automatisk. Dette er et svært spennende tema som prosjektets medlemmer samlet anbefaler sterkt å arbeide videre med. Dette er noe som også tas opp i *Kapittel 8 - Konklusjon (s. 55)*.

$$\begin{aligned} \text{Queries} &= Q = \{q_1, q_2, ...q_m\} \\ \text{Plugins} &= P = \{P_1, p_2, ...p_n\} \\ \text{Alarms} &= A = \{(q,p) | q \in Q \wedge p \in P\} \\ |A| &= |Q| * |P| \end{aligned}$$

Figur 16: Antall produserte alarmer

Grensesnitt

I et system hvor det skal utvikles mange plugins med et stort utvalg av forskjellige attributer de skal analysere er det viktig å utarbeide og opprettholde et felles grensesnitt for kommunikasjon mellom de enkelte plugins og kjernen av rammeverket som skal benytte seg av disse. En plugin i dette rammeverket kan analysere alt fra domenenavnentropi til antall numeriske tegn eller TLD-omdømme. Med utgangspunktet i datasettet som er levert til rammeverket (beskrevet i *Avsnitt 4.1.4 - CSV-format - Input (s. 22)*) er det store menger kombinasjoner og måter mistenklig aktivitet kan detekteres. Med dette i fokus - at systemet må kunne tilpasse seg nye metoder for deteksjon - skal hver plugin motta akkurat den samme dataen, og sende tilbake resultatet på en bestemt måte. Når datasettet om pDNS-aktivitetet leses inn fra fil eller som parametere vil dette lagres i - som tidligere beskrevet - et felles objekt som inneholder alle de forskjellige typene data man finner i slike pDNS-logger. Dette vil si at parameterene til en analysefunksjon i en plugin ikke vil måtte tilpasses for hver av de individuelle kravene den har til data, men at den heller tar imot et felles format, og er selektiv med sine informasjonsbehov internt i funksjonen.

Krav til plugins

For å ivareta en sømløs interaksjon mellom utviklede plugins og rammeverket er det satt visse krav til utformingen av alle plugins for å ivareta dette. Det første kravet er inputen som hver plugin skal ta imot. Mange plugins vil sannsynligvis ha forskjellige input-krav i form av hvilke DNS-attributter de ønsker å analysere for å gi vurderinger på. Ettersom dette kan som nevnt variere mye skal alle plugins ta imot en fullverdig DNS-oppføring gitt som et `QueryObject` (se *Avsnitt 4.2.4 - Queries (s. 31)*) og deretter trekke ut sine ønskede attributer for videre analyse. Hver analyseplugin skal ha en klasse med en funksjon ved navnet `analyze` som står for all analyse. Det er dog mulig å definere og utvikle så mange egne funksjoner man vil, men det er `analyze` funksjonen som vil bli kalt når en plugin kjøres i rammeverket. Et eksempel på en plugin sitt klassediagram ser man i *Figur 15 - Klassediagram for eksempelplugin (s.30)*. Opplever noen plugins feil eller for eksempelvis uforutsette dataformater skal de returnere en felles bestemt feilmelding (enten en bestemt verdi som -1 eller f.eks et `Error`-objekt i Python). Det har ikke blitt spesifisert noe nærmere hvordan denne feilhåndteringen skal være utformet ettersom det er usikkert om rammeverket skal videreutvikles og isåfall hvilket programmeringsspråk det vil være i. Dermed sees det som bedre å fremheve nødvendigheten til slik feilhåndtering kontra en detaljbeskrivelse av løsningen.

Etter en analyse er gjennomført skal den resulterende avgjørelsen eller matematisk resultat leveres tilbake som en `return` i form av et desimaltall mellom 0.0 og 10.0. Valget for tallvalget er løst basert på samtaler med oppdragsgiver og har ingen sterkere begrunnelse enn dette. Det har blitt vurdert å ha resultatet levert som en boolsk funksjon ved et `True/False` utfall på om den respektive analysen resulterer i en indikasjon på ondsinnet aktivitet eller ikke, men det ble tilslutt besluttet at rammeverket burde ha mulighet til å levere et mer nyansert bilde av resultatet fra hver analyse. Dog kan det fremdeles være mulig å behandle og begrense utfallet fra hver analyseplugin som boolske svar ved å definere 0 som `False` og 1 (eller en annen verdi) som `True`.

4.2.4 Queries

Som beskrevet tidligere i dette kapittelet så vil alle innleste pDNS-oppføringer som skal analyseres legges i et `Queries`-objekt. Overordnet er `Queries`-klassen en klasse som sty-

rer innlesning av loggfiler eller allerede gitte variabler, opprettelse av enkelte `Query`-objekter og tilgjengeliggjøring av disse enkelte objektene via en liste. I dette tilfellet falt valget på en såkalt *double ended queue (deque)* som vil gi rask aksess til både starten og slutten av listen. Dette er nyttig både for å legge inn nye objekter på slutten av listen, og for uthenting av de objektene som ble lagt inn først og da ligger i starten av listen.

Figur 17: Organisering av queries

Denne løsningen benyttes uansett om det er et eller ti milioner oppføringer som skal leses inn og analyseres. Grunnlaget for dette er å skape et felles grensesnitt slik at videre analysefunksjoner kun har et felles format og grensesnitt å forholde seg til med tanke på tilgang til pDNS-oppføringer. Dette sees også som en akseptabel løsning ettersom det er ønsket og beskrevet at rammeverket skal kjøres primært ved brukerinteraksjon. Dette fører til at det ikke kommer til å være et ressursproblem å lese inn alle oppføringer før analysen gjennomføres. Ved eventuell videreutvikling og benyttelse av rammeverket som en *stand-alone*-analyseenhet som kjører større datasett periodisk vil det være behov for å se på ytelsesforbedring ved f.eks å kutte ut `queries`-listen og kun forholde seg til enkelte `query`-objekter, og kjøre rett fra innlesning av oppføring videre til analyse og utskrivning av resultat.

Query-objekter

Som nevnt vil `Queries`-klassen håndtere en deque fylt med `Query`-objekter for hver oppføring som skal til videre analyse. I *Figur 7 - CSV-format for pDNS-loggfil (s.23)* er det definert og presentert de 8 forskjellige feltene en pDNS-logg som dette rammeverket skal kunne lese. Hvert av disse datafeltene vil fremstilles som individuelle variabler i et `query`-objekt sammen med funksjoner for å hente, endre og sette de respektive variablenes verdier.

Dette formatet som oppstår ved å bygge opp objektet og dens variabler møter de samme utfordringene som tidligere nevnt ved flere anledninger i dette kapittelet. Det låser låser seg til det gjeldende datasettet. En mulighet for videre utvidelse av denne klassen og dens funksjonalitet er å la brukeren - på lik linje og potensielt samme konfigurasjon som beskrevet i *Avsnitt 4.1.4 - CSV-format - Input (s. 22)* - konfigurere hvilke datafelter som denne klassen skal inneholde. Hvis det kun skal være et datasett er dette ikke sett som nødvendig bruk av utviklingstid, men hvis rammeverket i fremtiden skal kunne håndtere en større mengde forskjellige datasett vil dette være noe man burde vurdere. Samtidig så kommer det muligens en standard for pDNS-dataformat[3] (drevet av representanter fra den Østeriske CERTen) som det kan være interesant å tilpasse rammeverket til i fremtiden.

[3] `https://datatracker.ietf.org/doc/draft-dulaunoy-kaplan-passive-dns-cof`

5 Prototype

5.1 Rammer og avgrensning

5.1.1 Mål

Tidlig i prosessen av å avklare innholdet og målet med dette prosjektets oppgavetekst ble det fremmet et ønske om å utvikle en POC av rammeverket prosjektets medlemmer skulle identifisere behovene til. Ved å utvikle en proof-of-concept vil oppgaven bedre kunne illustrere og presentere det gjennomførte analyse og kartleggingsarbeidet med tanke på å indentifisere behovene til et rammeverk for analyse av pDNS data. Målet med dette kapittelet er å presentere, forklare og vurdere et POC-programvare som forsøker å implementere oppgavens tidligere utarbeidede alarmforslag og deres tilhørende analyserbare attributer.

5.1.2 Rammer

Oppdragsgiver ønsket at programvaren prosjektets medlemmer skulle utvikle ble utviklet i et programmeringsspråk som var mye brukt og godt kjent internt i mnemonic. Disse var programmeringsspråkene Python[1] og Perl[2]. For gruppens medlemmer var Python det programmeringsspråket som man hadde mest kjennskap om fra før. Python ble dermed valgt som programmeringsspråk for programvaren. Et viktig punkt for å velge Python var at dette ville medføre at det måtte settes av mindre tid til opplæring i et nytt språk.

Det ble ikke gitt eller satt noen føringer på krav eller eventuelle begrensninger på hva slags maskinvare eller operativsystemer prosjektets medlemmer kunne benytte under utviklingen. Ei heller ble det fremmet noen spesifikke krav med tanke på ressursbruk under kjøring av programmet. Ettersom programvaren som skal utvikles er en POC er det ikke tiltenkt at denne programvaren skal settes ut i produksjon. Evnetuelle slike feil, mangler eller ueffektive løsninger vil være elementer som må taes tak i og behandles under utvikling av en produksjonsversjon av rammeverket. Rammene for utviklings- og testmiljø hvilte da på at prosjektgruppen skulle benytte deres egne maskiner eventuelt datamaskiner tilgjengelig på skolen til vanlig bruk for studenter. En videre presisering av programmeringsspråk - herunder versjonsnummer og lignende - samt maskinevare og utviklingsmiljøer forklares ytterligere i *Avsnitt 5.1.4 - Avklaringer (s. 34)*.

Det ble vurdert som nevnt fra tidlig av hvor stor del av rapporten denne prototypen skulle besitte. Det falt på at dette er kun en prototype som fungerer som et POC for funnene til resten av oppgaven. Tyngden ligger i resultatene og drøftingene gjort i *Kapittel 3 - Deteksjon og analyse (s. 10)* og *Kapittel 4 - Kravspesifikasjon og design (s. 18)*. Orginalt var det tiltenkt en sprint (to uker) til utvikling av en prototype samt at dette skulle gjennomføres relativt sent i prosessen. Etterhvert ble det vurdert å ta denne prosessen tidligere samt tre uker ble satt av til utvikling.

[1] `https://www.python.org/about/`
[2] `http://www.perl.org/about.html`

5.1.3 Avgrensning

Med et prosjekt med et omfang og varighet som dette så vil man ikke kunne gjennomføre alt man har ønsker om, det viktigste må fremheves og eventuelle feil, mangler og avvik må begrunnes og dokumenteres. For forslag til videre arbeid med både arkitekturmessige og implementasjonsforbedringer se *Kapittel 6 - Videre arbeid (s. 49)*. Dette avsnittet tar for seg de avgrensningene det ble gjort med tanke på utviklingen og det som ikke ble utviklet i POC-programvaren. Først og fremst så er den produserte koden kun som nevnt en POC og ikke et system som skal settes inn i et produksjonsmiljø. Selv om dette er tilfellet er de kodekvalitetskravene som regnes som best-practices fulgt. Dette gjør at den produserte koden vil være lettere å bygge videre på, forstå samt kunne videreutvikle. Dokumentasjon og kodestil står da veldig sentralt. For dette har prosjektets medlemmer fulgt Python Enhancement Proposal Nr. 8[46] og Google Python Style Guide[47]. Rammeverket som er utviklet følger alle de gitte kravene satt i kravspesifikasjonen til det nivået man kan forvente fra en POC. Det har ikke blitt fulgt noen spesiell utviklingsmetodikk for selve programvaren enn at en og en modul ble produsert etter spesifikasjonene gitt i *Kapittel 4 - Kravspesifikasjon og design (s. 18)*. Det har ikke blitt gjennomført noen større ytelsestesting utenom en tidtakningstest som vist i *Figur 7 - Resultater ved testing av rammeverk (s.42)*. Det har heller ikke blitt gjennomført noen større optimaliseringsrevisjoner av koden utenom et mål å følge best-practice i utviklingen samt feilhåndtering når dette var nødvendig.

5.1.4 Avklaringer

Fore å best mulig kunne forstå eventuelle avvik eller miljømessige utfordringer vil det i dette avsnittet bli presentert avklaringer rundt gruppemedlemmenes utviklingsmiljø. Elementer som tas opp her er programmeringsspråket som ble benyttet til implementering av prototypen, maskinvare spesifikasjoner og andre punkter av potensiell interesse for å ha en mer komplett innsikt i utviklingen og prototypens rammer og avgrensninger.

Programmeringsspråk

Prototypen har blitt implementert i programmeringsspråket Python. Det var et ønske fra oppdragsgiver at rammeverket ble utviklet i dette eller Perl. Disse programmeringsspråkene er ofte brukt i mnemonic, og mange ansatte har god kjennskap til disse programmeringsspråkene. Gruppen har valgt å benytte seg av Python versjon '2.7.5'[3]. Python i sin nyeste versjon er '3.4.0' som ble utgitt 16. Mars 2014[48], men valget falt for den eldre versjonen. Det er flere grunner til dette. Først og fremst så er det versjonen som fulgte standard med prosjektmedlemmenes operativsystemer og var dermed enkelt tilgjengelig uten noen spørsmål om feil og mangler som kunne oppstå ved installering. Den andre grunnen til dette valget var at mange moduler og tredjepartsfunksjonaliteter ikke nødvendigvis vil fungere med Python versjon 3.x. Dette er ettersom ved utviklingen av versjon 3.x ble det besluttet å oppdatere programmeringsspråket og fikse feil og mangler uten å ta så mye hensyn til bakoverkompabilitet[49]. Videre så har gruppens medlemmer tidligere programmert - på hobby basis - i Python så dette ga en lavere terskel for å begynne arbeidet og eliminerte mye potensielt opplæringsarbeid, dermed kunne mer tid avsettes til andre deler av prosjektet.

[3] Python 2.7.5 (default, Aug 25 2013, 00:04:04) [GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.0.68)] on darwin

Maskinvare

1. **Andreas**: Apple Macbook Air 13"(2013 modell)
    1. Operativsystem: OS X 10.9.2 (13C64)
    2. Prosessor: Intel Core i7 @1.7 gHz (Haswell)
    3. Minne: 8 GB 1600 MHz DDR3
    4. Lagring: APPLE SSD SM0256F Media
5. **Lars**: Apple Macbook Pro 13"(2012 modell)
    1. Operativsystem: OS X 10.9.2 (13C64)
    2. Prosessor: Intel Core i7 2.9 Ghz
    3. Minne: 8 GB 1600 MHz DDR3
    4. Lagring: 256 GB Solid State SATA Drive

Versjonskontroll

Fra tidlig av bestemte gruppen seg for å benytte et versjonskontrollsystem. Det hadde ikke blitt gitt noen føringer eller krav til beskyttelse av koden under utvikling fra hverken oppdragsgiver eller gruppens medlemmer selv, så valget var fritt blant de muligheter som fantes. Valget for versjonskontroll havnet på Subversion (SVN). HiG leverer en SVN-løsning til deres studenter via `https://svn.hig.no/`, noe prosjektets medlemmer valgte å benytte seg av. Ved å gå for denne løsningen fremfor for eksempel tjenesten levert av GitHub[4] kunne prosjektets medlemmer lettere komme i kontakt med ansvarlige for løsningen samt prosessen med å komme i gang var svært lav. Det er ingen spesiell årsak med tanke på sikkerhet eller lignende til valg av SVN ovenfor for eksempel Git som versjonskontrollsystem.

Tildelt pDNS-Datasett

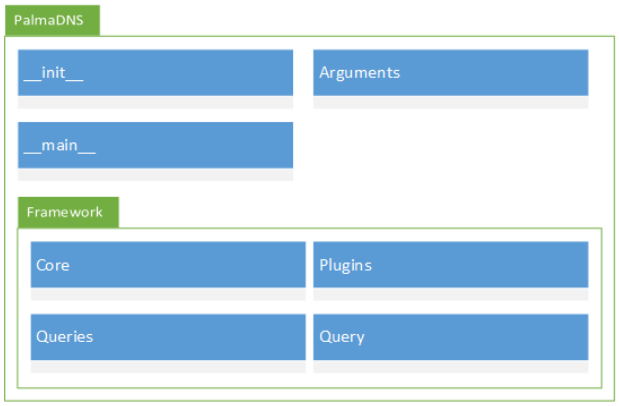
Under utviklingen av prototypen samt kravspesifisering av rammeverket og dens ønskede funksjonalitet hadde prosjektets medlemmer behov for et utdrag av data fra en pDNS-sensor fra vår oppdragsgiver mnemonic. Etter et møte med prosjektets kontaktperson i mnemonic - Andreas Bråthen - fikk prosjektets medlemmer utlevert et større datasett. Dette datasetter var på $\approx$ 1.4GB med litt over 15 millioner linjer (15,185,985) med en linje per oppføring slik beskrevet i *Kapittel 4.1.4 - CSV-format - Input (s. 22)*. Dette datasettet har blitt benyttet til testing av ytelse og ønsket funksjonalitet opp mot gitt kravspesifisering. Datasettet er delt ut til gruppens medlemmer merket med en TrafikkLysProtokollen (TLP)[50] nivå 'grønn' vurdering. Dette medfører at prosjektets medlemmer ikke kan publisere dette datasettet eller legge det til som vedlegg i denne rapporten. Beskivelsen gitt fra Norwegian Computer Emergency Response Team (NorCERT) ved Nasjonal Sikkerhetsmyndighet (NSM) lyder som følger med tanke på nivå 'grønn': "Informasjonen kan deles med andre virksomheter eller personer innen informasjonssikkerhetsmiljøet, men skal ikke publiseres eller legges ut på websider/åpne mailinglister."[50].

[4] `https://github.com/`

5.2 Struktur

5.2.1 Pakkestruktur

Fra starten av utviklingen var det ønsket å holde en oversiktlig struktur sammtidig som det var lett å tilpasse og å kunne gi nye brukere utvidede muligheter for bruk og videreutvikling av rammeverket. Som prosjektets medlemmer ser i *Figur 19 - Implementert filstruktur (s.40)* er rammeverket først i en mappe med navnet 'PalmaDNS'. Denne overordnede mappen inneholdet informasjon om programmet som README.txt (kjapp bruksanvisning og informasjon om programmet) samt LICENSE.txt (lisens som programmet er gitt ut under). Videre så inneholder den to python filer. '__init__.py' og '__main__.py'. Init-filen er til for at den overordnede mappen skal kunne fungere som en 'package' i python programmeringsspråket. En package i Python er en måte å strukturere python-modulers namespace på. For eksempel i dette tilfellet vil man kunne importere PalmaDNS og kalle filen '__main__.py' ved bruk av 'PalmaDNS.__main__'. Dette forenkler prosessen av å potensielt ha like navn som andre moduler i Python språket samtidig som det tilbyr en svært oversiktlig og strukturert organisering av programmets kildekode. Et overordnet diagram over denne strukturen vises i *Figur 18 - Overordnet pakkestruktur (s.36)*.



Figur 18: Overordnet pakkestruktur

'__main__.py' i hovedmappen lar brukeren kalle mappen 'PalmaDNS' direkte som et kjørbart script. Python merker at det er en mappe som har blitt gitt som kommandolinjeargument og ser deretter etter fil med navnet '__main__.py'. Et eksempel på slik kjøring ser man i *Figur 6 - Eksempler på kjøring av rammeverket via kommandolinje (s.22)*. Videre er også filen 'arguments.py' på dette nivået av kildekoden. Dette er siden 'arguments.py' arbeider med behandling og validering av kommandolinjeargumenter. Dette er ikke en sentral del i selve rammeverket ettersom det ikke bidrar til analyse eller lignende, og er dermed lagt sammen med '__main__.py' filen. Oppsumert sett kan dette overordnede nivået sees på som hjemmeområdet for alt av filer som skal til for at rammeverket skal kunne kjøres i henhold til kravene satt i *Kapittel 4 - Kravspesifikasjon og design (s. 18)* fra et CLI-miljø. Videre er resten av rammeverket lagt i undermappen 'Framework'. Denne mappen inneholder en '__init__.py' fil som gjør at den behandles som en package av Python. Det vil si at for å benytte for eksempel Core-objektet vil man kunne skrive 'import Core from PalmaDNS.Framework.core' i Python. Denne inndelingen er valgt slik at alle moduler som direkte har noe med selve rammeverket å gjøre - da med tanke på plugins, analyse, queries, og lignende - befinner seg i 'PlamaDNS.Framework' pakken, mens resterende funksjonalitet som berører brukerinteraksjon og informasjon om rammeverket befinner seg i den overordnede 'PalmaDNS' mappen.

5.2.2 Klasser

Under utviklingen av prototypen har det blitt beholdt de orginale navnene og inndelingen av funksjonalitet som beskrevet i *Kapittel 4 - Kravspesifikasjon og design (s. 18)* for bedre kunne vise sammenhengen mellom utpekte ønsker til krav og design, og selve implementasjonen. I *Appendix A - Figurer og diagrammer (s. 60)* har det blitt utarbeidet to UMLklassediagrammer som presenterer de forskjellige klassene og delt opp i deres respektive python packages.

Appendix A.4 - Klassediagram for en Python-implementasjon av PalmaDNS (s. 63) viser UML-klassediagrammet til den overordnede packagen 'PalmaDNS' som beskrevet inneholder klassen for Arguments og en '__main__.py' for kjøring av rammeverket. '__main__.py' er spesiel i det at når en bruker vanligvis kjører et python program designerer de hvilken fil som skal kjøres, men med en '__main__.py' fil i et direktoray kan man kalle den mappen og python vil automatisk kjøre denne filen. Denne '__main__.py' filen initierer og kjører rammeverket ut i fra gitte instruksjoner fra brukeren via kommandolinjeargumenter som blir håndtert fra 'Arguments'-klassen.

Appendix A.5 - Klassediagram for en Python-implementasjon av Framework-komponenter (s. 64) viser UML-klassediagrammet til alle klasser som befinner seg i 'PalmaDNS.Framework' packagen. Dette er som beskrevet i *Kapittel 4 - Kravspesifikasjon og design (s. 18)* Core, Queries, Plugins og Query. I tillegg til disse klassene inneholder Query-klassen en nestet klasse kalt 'QueryData'. Dette er gjort ettersom Query objektene inneholder mange variabler som skal ha samme funksjonalitet. Denne felles funksjonaliteten er muligheten til å sette verdien til variablene og hente ut verdien til variablen. I stedet for å lage både 'hente verdi'- og 'sette verdi'-funksjoner til hver variabel ble det laget denne underklassen som heter 'QueryData' for å levere nettopp denne funksjonaliteten.

Forklaring til klassediagrammer

- Alle klasser er tredelt med stripplede avgrensende linjer. Den øverste sektoren er for interne variabler. Neste sektor er for private funksjoner. Den siste sektoren er for offentlige funksjoner. Noen av klassene vil muligens ikke inneholde noen private variabler eller funksjoner og dermed kan disse være unnlatt der det er behov. Et eksempel på dette ser man i *Appendix A.5 - Klassediagram for en Python-implementasjon av Framework-komponenter (s. 64)* hvor 'Query klassen ikke innehar noen offentlige funksjoner'.
- Selv om alle funksjoner i en python klasse er offentlige uansett er det regnet som best-practice å betegne private variabler og funksjoner med to understreker før navnet '__'[47].
- funksjoner med '@' foran navnet er her i klassediagrammet en forkortelse av en eller flere funksjoner som omhandler python property decorators[51].

5.2.3 Eksterne moduler

For å kunne levere mer av funksjonaliteten som har blitt etterspurt via kravspesifikasjonen ble det nødvendig i utviklingen av prototypen å benytte seg av eksterne moduler. Med eksterne moduler her så menes moduler til programmeringsspråket Python som leverer mer enn det Python alene kan gjøre ut-av-boksen. Mange av disse modulene er levert som en del av Python distribusjoner. Nedenfor vil listes disse opp samt en forkla-

ring for behovet presenteres.

- 'logging'
  - Beskrivelse: 'logging'-modulen er et fleksibelt hendelsesloggføringssystem for applikasjoner og biblioteker.
  - Benyttet i prototypen: 'logging' blir benyttet i de fleste moduler av rammeverket for å tilby alt fra verbose debugging utskrift hvis brukeren har ønsket det, til utskrift av feilmeldinger hvis og når disse oppstår. Brukeren har også mulighet ved logging.ini-konfigurasjonsfilen i rammeverket til å endre flere detaljer ved loggføringen, som for eksempel at utskriften skal sendes til en fil kontra til skjerm.
  - Dokumentasjon: `https://docs.python.org/2/library/logging.html`
- 'os'
  - Beskrivelse: 'os'-modulen gir python tilgang til en rekke forskjellige Operativsystem (OS) funksjonaliteter uavhengig av operativsystem, på lik linje med python sin mulighet å fungere likt på flere forskjellige platformer. Noen av funksjonene er da kun tilgjengelig på OS-typene hvor slik funksjonalitet er vanlig, et eksempel på dette er 'os.getgid()' som kun er tilgjengelig på 'Unix'.
  - Benyttet i prototypen: Denne modulen benyttes i '__main__.py' og 'core.py' for å hente ut fullstendige stier, altså ikke relative stier som f.eks '../file.test' men heller '/var/log/file.test'. I '__main__.py' benyttes det for å finne den fullstendige stien til 'logging' modulen sin konfigurasjons fil, og i 'core.py' brukes det for å hente den fullstendige stien til mappen hvor rammeverkets plugins ligger.
  - Dokumentasjon: `https://docs.python.org/2/library/os.html`
- 'ConfigParser'
  - Beskrivelse: 'ConfigParser'-modulen tilbyr en enkel og effektive konfigurasjonsfilparser og behandler. Den lar deg håndtere konfigurasjonsfiler med en Microsoft Windows INI fil type struktur.
  - Benyttet i prototypen: Denne modulen blir benyttet kun i '__main__.py' modulen. Her blir den benyttet til å sette og sjekke det nåværende nivået (debug, info, warning, etc) som loggføringskonfigurasjonen er satt til. Hvis brukeren har ønsket utskrift av debugginginformasjon mens konfigurasjonen er satt til å kun viser 'warning' nivå vil dette endres.
  - Dokumentasjon: `https://docs.python.org/2/library/configparser.html`
- 'argparse'
  - Beskrivelse: 'argparse' er en modul som gjør det enkelt å kunne utvikle kommandolinjeargumentgrensesnitt og menyer til et program for at en bruker lettere skal kunne benytte seg av et program med varierende funksjonalitet.
  - Benyttet i prototypen: Denne modulen blir kun benyttet i 'arguments.py'-modulen, og mer spesifikt i 'Arguments' klassen. Det meste i 'arguments.py' baserer seg rundt 'argparse' ettersom det er rundt denne modulen som menysystemet til rammeverket, som gir brukeren mulighet til å velge ønsket funksjonalitet og handling

fra rammeverket.

  - Dokumentasjon: `https://docs.python.org/2/library/argparse.html`

- 'collections'
  - Beskrivelse: 'collections'-modulen tilbyr en rekke datastrukturer som skal være direkte alternativer de de innebygde datatypene til python som f.eks 'dict', 'list', 'set' og 'tuple'.
  - Benyttet i prototypen: Denne modulen blir benyttet i 'queries.py' modulen hvor det blir opprettet en 'collections.deque()' (double-ended queue) for å holde på samtlige pDNS-oppføringer.
  - Dokumentasjon: `https://docs.python.org/2/library/collections.html`
- 'yapsy'
  - Beskrivelse: 'yapsy' er et enkelt lettvektssystem for implementering av et system for utvikling og håndtering av plugins.
  - Benyttet i prototypen: Denne modulen blir benyttet i hver plugin som er utviklet samt i modulen 'plugins.py' (som står for håndtering av samtlige plugins). 'yapsy' kan enkelt deles opp i to klasser som tilbyr dens funksjonalitet. Først har man klassen 'PluginManager' (benyttes i 'plugin.py' som tilbyr all plugin-håndtering (aktivering, uthenting av navn og informasjon samt kjøring av plugins). Videre har man klassen 'IPlugin' (hver utviklet plugin sin klasse arver fra denne) som definerer en plugin sitt grensesnitt som håndteres av 'PluginManager'.
  - Dokumentasjon: `http://yapsy.sourceforge.net/`

Denne listen inkluderer kun den importerte funksjonaliteten som er benyttet i utviklingen av selve rammeverket og dens kode. Eksterne moduler som er benyttet i de enkelte plugins er beskrevet i deres respektive forklaringer i *Avsnitt 5.4 - Plugins (s. 42)*.

5.2.4 Filstruktur

Den implementerte filstrukturen til rammeverket vises i *Figur 19 - Implementert filstruktur (s.40)*. Denne implementeringen er en videreføring av tankegangen til forslaget til filstrukturen beskrevet i *Avsnitt 4.2.2 - Filstruktur (s. 29)*. Som nevnt *Avsnitt 5.2.3 - Eksterne moduler (s. 37)* er modulen Yet Another Plugins SYstem (Yapsy) valgt til håndtering av plugins, derfor er det en '*.yapsy-plugin' konfigurasjonsfil per utviklet plugin. All funksjonalitet kyttet til bruk av rammeverket som en selvstendigenhet ligger i den overordnede mappen '\PalmaDNS' mens selve analyse funksjonaliteten er i undermappen '\PalmaDNS\Framework'.

5.3 Funksjonalitet

5.3.1 Brukerinteraksjon

En fullstendig oversikt over alle kommandolinjeargumenter som rammeverket skal kunne ta imot for å gi brukeren tilgang til ønsket funksjonalitet er beskrevet i *Avsnitt 4.1.3 - Kommandolinjeargumenter (s. 20)*. Samtlige av disse kommandolinjeargumentene er implementert i prototypen av rammeverket. Som beskrevet i *Avsnitt 5.2.3 - Eksterne moduler (s. 37)* benytter rammeverket Python modulen 'argparse' til håndtering av kommando-

```
\PalmaDNS
|---- LICENSE.txt
|---- README.txt
|---- __init__.py
|---- __main__.py
|---- arguments.py
|---- \Framework
|-----|---- __init__.py
|-----|---- core.py
|-----|---- plugins.py
|-----|---- queries.py
|-----|---- query.py
|-----|---- \plugins
|-----------|---- geolocation.py
|-----------|---- geolocation.yapsy-plugin
|-----------|---- ttl.py
|-----------|---- ttl.yapsy-plugin
|-----------|---- sample_plugin.py
|-----|---- \config
|-----------|---- logging.ini
```

Figur 19: Implementert filstruktur

linjeargumenter. 'Argparse'-modulen håndterer hvis brukeren gir feil format på input eller potensielt ugyldige argumenter. For å gjøre kildekoden mer oversiktlig har funksjonaliteten for konfigurering og behandling av kommandolinjeargumenter blitt skilt ut av '__main__.py' og blitt lagt i filen 'arguments.py' i klassen 'Arguments'. *Figur 20 - Visning av gyldige parametere / hjelpmeny i rammeverket (s.40)* viser den automatiske genererte hjelpemenyen for å assistere brukeren til å finne frem til ønsket funksjonalitet. Denne menyen blir produsert og håndtert av modulen 'argparse' etter at ønskede parametere er konfigurert, noe som har spart mye utviklingstid.

```
Andreass-MacBook-Air-2:Code andmoe$ python PalmaDNS run params -h
usage: PalmaDNS run params [-h] -q QUERY [-a ANSWER] [-t TTL] [-p N [N ...]]
                           [-o OUTPUT] [-f] [-v]

optional arguments:
  -h, --help            show this help message and exit
  -q QUERY, --query QUERY
                        DNS Query
  -a ANSWER, --answer ANSWER
                        Query answer
  -t TTL, --ttl TTL     Time To Live
  -p N [N ...], --plugins N [N ...]
                        Wanted plugin(s)
  -o OUTPUT, --output OUTPUT
                        Output file
  -f, --formated        Shows fancy output to screen
  -v, --verbose         Shows verbose debuging information
ERROR - PalmaDNS experienced an error and was exited
Exiting...
Andreass-MacBook-Air-2:Code andmoe$ _
```

Figur 20: Visning av gyldige parametere / hjelpmeny i rammeverket

Validering av input

Som tidligere nevnt vil Arguments-klassen håndtere eventuelle feilaktige kommandolinjeargumentoperatører, som eksempel hvis en bruker gir en input fil '–input test.log' når brukeren ønsker å analysere parametere. Utenom sjekking av gyldige og tillate parametere er det ingen annen sjekking av gyldigheten til input gitt fra brukeren. Med dette så menes det at verdien til de forskjellige parameterene - eksempelvis domenenavn, TTL, IP-addresser - vil ikke bli sjekket for å se om de er faktisk av et gyldig format i henhold til forventet. Med dette så kan en bruker sende med feilaktige verdier samt potensielt avdekke sårbarheter som kan forstyrre programmets funksjonalitet. Ettersom dette er en prototype har det blitt vektlagt å produsere kildekode som oppfyller kravene satt til funksjonalitet i *Kapittel 4 - Kravspesifikasjon og design (s. 18)*. I et rammeverk som skal settes inn i et produksjonsmiljø er det sterkt anbefalt å implementere en sikker og grundig validering av brukermedsendt data. For en slik validering er det to kategorier av feil som burde kunne detekteres:

1. Ondsinnet input: Input gitt fra en bruker med ondsinnede intensjoner. Dataen som blir gitt som input her vil kunne - alt ettersom hvilken sårbarhet i rammeverket, Python eller andre mulige sårbare systemer - benytte seg av rammeverkets inputfunksjonalitet til å utgjøre skade, eller forstyrre systemet rammeverket opererer på det. Konsekvenser kan i ytterstefall være datalekkasje, ødeleggelse eller forstyrrelse av andre prosesser.
2. Feilaktig input: Input gitt fra en bruker som kan produsere samme eller lignende feil som en ondsinnet bruker, men uten at det er en bevist handling. Videre kan dette også være data som generelt blir ansett ikke gyldig med tanke på rammeverkets ønsket pDNS-format. Dette kan gi feilaktige eller ugyldige analyseresultater og eller resultere i ustabilitet i kjøringen av rammeverket.

5.3.2 Resultater

I *Avsnitt 4.1 - Kravspesifikasjon (s. 18)* ble det satt visse krav og mål for en implementasjon av rammeverket. Denne implementasjonen tilbyr en rask og oversiktlig måte for analytikere å analysere pDNS. Alle funksjonelle krav er dekket og den støtter de kravene som er satt for output og input. Det har blitt utviklet to plugins som presenteres i *Avsnitt 5.4 - Plugins (s. 42)* og rammeverket er implementert slik at det kan benyttes i andre løsninger og programmer. Dette er presentert i *Avsnitt 5.5 - Bruk av rammeverket i andre programmer (s. 44)*. Prototypen er pluginbasert og designet bygger på de samme prinsippene som er presentert i *Avsnitt 4.2.1 - Moduler (s. 27)*. Implementasjonen er klar for at plugins kan implementeres og inkluderes i rammeverket.

```
>> time python PalmaDNS run file -i pdns_1000.log -o results.csv
```

Figur 21: Kjøring av tidstest

For å vurdere ressursbruk og ytelse av rammeverkets prototype har det blitt gjennomfører en rekke systematiske tidtakninger med varierende mengde input data. Disse testene er utført ved hjelp av kommandoen i *Figur 21 - Kjøring av tidstest (s.41)*, her vist for 1000 linjer med pDNS. Når denne kommandoen blir kjørt vil rammeverket gjennomføre en fullstendig oppstart med innlesning av kommandolinjeargumenter, initialisering av rammeverket, innlesning av datasett samt utskrift av resultater. Her

det ikke satt noen begrensning på hvilke eventuelle plugins som skal kjøres. I dette tilfellet kjørte plugins relatert til TLD og TTL, ettersom disse var de eneste pluginene lagt inn i rammeverket for dette test scenarioet. Se *Avsnitt 5.4 - Plugins (s. 42)* for mer informasjon og disse pluginene. Disse 7 testene er alle utført på Andersen sin maskin, se *Avsnitt 5.1.4 - Maskinvare (s. 35)* for maskinvarespesifikasjoner. Testene er utført på utdrag av dataen presentert i *Avsnitt 5.1.4 - Tildelt pDNS-Datasett (s. 35)*. *Tabell 7 - Resultater ved testing av rammeverk (s. 42)* viser resultatene av disse testene. Her har antall pDNS-queries vært i spekteret 1 til 1,000,000 med en økning på 10 ganger så mye ved hver nye test. Dette ble valgt for å undersøke hvordan rammeverket vil håndtere både mindre menger data som kan være håndterlig for en analytiker uten andre verktøy til videre analyse, samtidig som det blir undersøkt For å få et mer realistisk resultat har hver test blitt kjørt totalt 5 ganger og resultatene presentert her er et gjennomsnitt av disse.

Antall queries	Tidsbruk i sekunder
1	0.0874
10	0.0886
100	0.0946
1,000	0.1492
10,000	0.7494
100,000	7.0280
1,000,000	69.6095

Tabell 7: Resultater ved testing av rammeverk

5.4 Plugins

Som beskrevet i listen over eksterne moduler i *Avsnitt 5.2.3 - Eksterne moduler (s. 37)* ble det tatt et valg om å bruke Yapsy til å gjøre prototypens analysefunksjonalitet basert på en plugin-struktur. Yapsy gjør arbeidet med å lage plugins både enkelt og elegant. Ved å tilby to kjerneklasser til arbeid med plugins oppnår Yapsy at det å laste inn og aktivere plugins er enkelt. Alle nye plugins må lages i `\PalmaDNS\Framework\plugins` og det må skrives en konfigurasjonsfil til hver plugin. Et eksempel på en slik konfigurasjonsfil vises i vedlegget *Avsnitt B.11 - /PalmaDNS/Framework/plugins/tld.yapsy-plugin (s. 80)*. Det viktigste i en slik fil er å få med de tre første linjene som definerer navnet som skal brukes for å kalle pluginen og hvilken pythonfil som skal kjøres, her henholdsvis "001-TLD"og tld". Pluginene som er utviklet her er kun for å vise at rammeverket fungerer slik det er forventet, og det har blitt tatt et valg om å simplifisere tilbakemeldingen hver plugin gir. Hver av de implementerte pluginene vil returnere enten 0.0 hvis den klassifiserer domenet som godsinnet og 10.0 hvis den klassifiserer domenet som ondsinnet. Som nevnt i *Avsnitt 4.2.3 - Krav til plugins (s. 31)* skal derimot pluginene i produksjon returnere en desimalverdi mellom 0.0 og 10.0.

5.4.1 TLD

Implementasjon

Implementasjonen av TLD-pluginen er relativt simple. Som nevnt i *Avsnitt 3.2.2 - (8)TLD (s. 16)* ble det foreslått en liste over TLD-er som potensielt kan være risikofylte. Her ble det også nevnt at denne listen ikke har faglig tyngde nok til å brukes i produksjon, men kun generert for denne prototypen ved hjelp av data som er presentert i denne rapporten. For å hente ut TLD fra URL-en ble pythonmodulen Tldextract[5] utviklet av John Kurkowski brukt. Denne gjør prosessen for å hente ut de forskjellige elementene av en

[5]https://github.com/john-kurkowski/tldextract

URL meget enkel. Ved hjelp av Tldextract blir alle TLD-er sjekket opp mot en predefinert liste over potensielle risikofylte TLD-er og får enten en verdi på 10.0 eller 0.0 ettersom den henholdsvis finnes i listen eller ikke. Denne koden inneholder orginalt flere kommentarer og andre elementer som er nødvendig for fullstendig funksjonalitet, men dette er fjernet for plassbesparelse, se *Appendix B.10 - /PalmaDNS/Framework/plugins/tld.py (s. 80)* for utfyllende informasjon.

```
1  class TLD(IPlugin):
2      def analyze(self, qobj):
3          # List over potentialy malicious TLDs
4          tld_list = ['info', 'vn', 'cm', 'ms', 'am', 'com.co']
5          # Extract and find the TLD for this query
6          queried_domain = qobj.query.value
7          ext = tldextract.extract(queried_domain)
8          tld = ext.suffix
9
10         # Determin result of analysis
11         if tld in tld_list:
12             return 10.0
13         else:
14             return 0.0
```

Figur 22: Implementasjon av TLD-plugin

Resultater

Som nevnt i *Avsnitt 5.1.4 - Tildelt pDNS-Datasett (s. 35)* som har gruppen fått tildelt en pDNS av mnemonic. Denne inneholder tilsammen 15,185,985 linjer med pDNS. Ved kjøring av PalmaDNS med kun TLD-pluginen aktivert resulterer det med 66,848 ($\approx$ 0.00440%) linjer som trigget testen og fikk dermed 10.0 i verdi. Prosjektgruppen har fått tillatelse fra dataeier til å legge ut tilfeldige mindre utdrag som ikke har noe sammenheng.

```
195.159.140.196,60.190.217.130,wikis.ovear.info.,001-TLD,10.0
195.159.140.196,65.55.92.152,mx2.hotmail.com.,001-TLD,0.0
195.159.140.196,68.1.17.3,mx.east.cox.net.,001-TLD,0.0
195.159.140.196,93.94.216.162,www.panorama.am.,001-TLD,10.0
195.159.140.196,79.98.27.236,www.businesslist.vn.,001-TLD,10.0
195.159.140.196,144.76.86.115,budbad.com.,001-TLD,0.0
```

Figur 23: Utdrag av resultat fra TLD-plugin

I *Figur 23 - Utdrag av resultat fra TLD-plugin (s.43)* så er 6 tilfeldige linjer trukket ut fra dette resultatet, tre som trigget og tre som ikke trigget. Manuell analyse i etterkant viser at resultatet er riktig, og dette viser at rammeverkets interaksjon med plugins fungerer som det skal.

5.4.2 TTL

Implementasjon

For å differensiere godsinnede og ondsinnede domener ved hjelp av TTL brukes data som ble presentert i *Avsnitt 3.2.2 - (7) TTL-verdi (s. 15)*. Her nevnes det at det ble sett mange TTL-verdier i spekteret [0, 100] og det er dermed valgt å bruke 100 som terskel i denne implementeringen. Igjen så er ikke dette en godkjent verdi for bruk i produksjon men heller for å vise at rammeverket fungerer som det skal. *Figur 24 - Implementasjon av TTL-plugin (s.44)* viser en simpel test for TTL. Her vil alle linjer som har en TTL mindre eller

lik 100 bli gitt 10.0 i verdi, og alle andre 0.0. Denne koden inneholder orginalt flere kommentarer og andre elementer som er nødvendig for fullstendig funksjonalitet, men dette er fjernet for plassbesparelse, se *Appendix B.12 - /PalmaDNS/Framework/plugins/ttl.py (s. 80)* for utfyllende informasjon.

```
1  class TTL(IPlugin):
2      def analyze(self, qobj):
3          result = 0.0
4
5          # TTL value may not be set so a try/except block is needed
6          try:
7
8              # Extract and analyze the TTL value of this query
9              ttl = int(qobj.ttl.value)
10             if ttl <= 100:
11                 result = 10.0
12
13         # If no TTL value was set in this query
14         except ValueError:
15             pass
16         return result
```

Figur 24: Implementasjon av TTL-plugin

Resultater

Ved kjøring av PalmaDNS med kun TTL-pluginen aktivert på den tidligere nevnte pDNS-loggen resulterer det i 2,574,176 ($\approx$ 0.16951%) linjer som trigget testen og fikk 10.0 (utslag på en ondsinnet TTL $\leq$ 100) i verdi.

```
195.159.140.196,203.205.160.43,mx3.qq.com.,002-TTL,0.0
195.159.140.196,98.138.112.32,mta5.am0.yahoodns.net.,002-TTL,0.0
195.159.140.196,183.60.187.44,ara.sina.com.cn.,002-TTL,10.0
10.5.15.30,54.225.194.93,247026106.log.optimizely.com.,002-TTL,10.0
10.5.15.60,184.73.233.134,log.dmtry.com.,002-TTL,10.0
10.5.20.111,54.230.46.37,magasinsurance.com.,002-TTL,0.0
```

Figur 25: Utdrag av resultater fra TTL-plugin

I *Figur 25 - Utdrag av resultater fra TTL-plugin (s.44)* så er igjen seks tilfeldige linjer trukket ut fra dette resultatet, tre triggere og tre som ikke trigget. Manuell analyse i etterkant viser at alle linjer som ble trigget hadde en TTL-verdi på 100 eller mindre.

5.5 Bruk av rammeverket i andre programmer

Rammeverket som har blitt presentert i de tidligere avsnittene i dette kapittelet fungerer utmerket og i tråd med krav satt i *Kapittel 4 - Kravspesifikasjon og design (s. 18)*, som en selvstendig applikasjon en analytiker kan benytte seg av fra sin arbeidsstasjon i et CLI-miljø. Dette var også det primære målet med rammeverket med tanke på brukerinteraksjon. Selv om dette kravet er oppfylt ble det sett som ønskelig å presentere et utvalg av mulige utvidelser samt andre måter å benytte rammeverket på. Denne egenskapen ved rammeverket vil bli presentert i de kommende to avsnittene. Først en presentasjon over hvordan rammeverket kan benyttes i et Bash Script. Deretter presenteres et minimalistisk web-grensesnitt bygget med Python-rammeverket Flask.

5.5.1 Bash wrapper

Når rammeverket gjennomfører en analyse lastes alle de ønskede pDNS-oppføringene inn i rammeverket til 'queries.py' modulen sin deque. Som forklart i *Avsnitt 4.2.4 - Queries (s. 31)* vil dette fungere bra opp mot det primære formålet til rammeverket, nemelig en analytiker som skal vurdere et mindre datasett om gangen. Hvis ene analytiker ønsker å laste et større datasett inn i rammeverket kan dette by på utfordinger med tanke på ressurser i datamaskinen som skal kjøre analysen. En pDNS-logg med flere milioner oppføringer setter store krav til minne. En løsning til dette problemet er å dele opp en stor pDNS-logg til flere mindre filer som kan analyseres sekvensielt. Dette gjør at mindre data av gangen må lastes inn i minne for analyse, samtidig som resultater vil være raskere tilgjengelig enn å måtte vente til hele datasettet er ferdig analysert. *Appendix C.5 - /large_file_analysis.sh (s. 141)* viser kildekoden til et slikt script. Dette bash-skriptet tar en pDNS-loggfil som første argument etter kallet til skriptet. Deretter vil denne filen deles opp i 100,000 linjer lange filer. Det har ikke blitt gjort noen undersøkelser på den mest optimale størrelsen for fillengder. Dette kan være et område for videre undersøkelser for å eliminere mest av overheaden skapt ved å starte rammeverket på nytt gjentatte ganger samtidig som man har små nok filer i forhold til ressurser som er tilgjengelige på datamaskinen dette skriptet kjøres på. Videre så kjøres hver av de oppdelte filene i en instanse av rammeverket som sett på linje

```
>> ./large_file_analysis.sh pdns.log
```

Figur 26: Kjøring av bash wrapper-skript

```
Andreass-MacBook-Air:Code andmoe$ ./large_file_analysis.sh input/pdns.log.1m

+-------------------------------------------------------+
|       Large file analysis for PalmaDNS                |
+-------------------------------------------------------+

        - Will analyze 1000000 lines of pDNS data
        - Split into 10 file(s)
        - Start: 21:03:43

        + Started analysis of tmp_aaa...  [ DONE ]
        + Started analysis of tmp_aab...  [ DONE ]
        + Started analysis of tmp_aac...  [ DONE ]
        + Started analysis of tmp_aad...  [ DONE ]
        + Started analysis of tmp_aae...  [ DONE ]
        + Started analysis of tmp_aaf...  [ DONE ]
        + Started analysis of tmp_aag...  [ DONE ]
        + Started analysis of tmp_aah...  [ DONE ]
        + Started analysis of tmp_aai...  [ DONE ]
        + Started analysis of tmp_aaj...  [ DONE ]

        - Finnished: 21:04:25
        - Results are in ./results/tmp_*.results
```

Figur 27: Kjøring av large_file_analysis.sh med 1 milion queries

20 i *Figur 26 - Kjøring av bash wrapper-skript (s.45)*. For at brukeren skal kunne bedre følge med på handlinger som skriptet foretar seg samtidig som fremgangen vises mer eksplisit er det lagt inn flere linjer i scriptet som kun omhandler stilsetting og utskrift av informasjon til brukeren. Elementer her er for eksempel hvor mange filer den orgina-

le loggen ble delt i samt når analyse for de enkelte del-filene er ferdige. Tidspunkt for start og stopp av hele kjøringen vises også. Et eksempel på en kjøring av dette skriptet gjennomført på datamaskinen til gruppemedlem Andreas Moe vises i *Figur 27 - Kjøring av large_file_analysis.sh med 1 milion queries (s.45)*. Som man ser i skriptet er det ikke foretatt noen form for vurdering om filen legitimitet uten om en sjekk med operatøren '-e' som leverer 'True' om angitt fil eksisterer. Det er opp til rammeverket å vurdere videre validitet og gi de riktige feilmeldinger deretter. En slik wrapper for kjøring av større loggfiler kunne også vært skrevet i Python hvor rammeverket kunne vært importert som en modul.

5.5.2 Webgrensesnitt via Flask

Rammeverkets protoype er utviklet i Python med en modul- og pakkestruktur som gjør at det lett kan importeres i andre python-programmer. Dette gjør at rammeverket vil kunne benyttes direkte i nye løsninger og andre allerede utviklede programmer. Flask[52] er et såkalt mikrorammeverk for hurtig utvikling av websider med bruk av python og html templates. Flask er utviklet i Python noe som gjør at denne rapportens utviklede prototype lett kan importeres og benyttes fullt ut i det nye Python-programmet for å generere et web-grensesnitt. Hvordan dette gjøres kan man se på linje 18 i kildekoden for web-grensesnittet i *Appendix C.4 - /Web/web_ui.py (s. 140)*.

Figur 28: HTML-skjema for input av en enkelt pDNS-oppføring

Figur 29: Hjemmeside som viser resultater fra analyse i rammeverket

Ved å ha rammeverket tilgjengelig via et webgrensesnitt har utviklerene og leverandørene - eksempelvis systemadministratorer og lignende - muligheten til å sentralisere hvor alle plugins ligger og gjøre de nyeste plugins tilgjengelige for flere analytikere samtidig uten å måtte distribuere ny kode rundt. Videre vil dette også sentralisere behovet for ressurser til en slik analyse. Hvis det er større datasett som skal analyseres vil en kraftigere sentralisert server være nyttigere enn at de enkelte analytikerene skal låse opp sine ressurser til en slik handling.

Samtidig så kan et webgrensesnitt potensielt senke læringskurven for nye analytikere ved å tilby et grensesnitt som er intuitivt og ergonomisk designet[6]. I tillegg til dette vil også en slik sentralisert løsning gjøre at de enkelte datamaskinene til analytikeren ikke trenger å bry seg om ressurskrav utenom det å kjøre en nettleser med tilgang til serveren som innehar rammeverket og dens webgrensesnitt. Filstrukturen til dette grensesnittet vises i *Figur 30 - Filstruktur for webgrensesnitt (s.47)*. Cascading Style Sheets (CSS)-filene er henholdsvis: 'bs.css' - Bootstrap[7] og 'fui.css' - Flat-UI[8]. Kildekoden for HTML og Python filen(e) finnes i *Appendix C - Kildekode for web-grensesnitt og andre skripts (s. 139)*.

```
\Web
|---- web_ui.py
|---- \templates
|-----|---- layout.html
|-----|---- params_form.html
|-----|---- results.html
|---- \static
|-----|---- bs.css
|-----|---- fui.css
```

Figur 30: Filstruktur for webgrensesnitt

Ved bruk av et slikt rammeverk kan man lage flere spesial tilpassede sider for de forskjellige formene for analyse. Det kan være funksjonalitet for å laste opp filer, søke igjennom tidligere analyser (for å spare ressurser ved å unngå gjentagende analyse), eller enkle søk på individuelle pDNS-oppføringer. *Figur 28 - HTML-skjema for input av en enkelt pDNS-oppføring (s.46)* viser et eksempel på et grensesnitt for input av individuelle oppføringer og *Figur 29 - Hjemmeside som viser resultater fra analyse i rammeverket (s.46)* viser hvordan resultater fra en analyse potensielt kan bli presentert til brukeren.

5.6 Dokumentasjon

For at det skal være enklere å sette seg inn i koden til PalmaDNS, og dermed gjøre eventuell videreutviklingsarbeid lettere har det blitt generert dokumentasjon for kildekoden til det utviklede rammeverket, som et ekstra produkt ved siden av prototypen. Denne dokumentasjonen er generert ved hjelp av programmet Doxygen[9], og er å finne i *Appendix B.17 - Dokumentasjon for PalmaDNS (s. 82)*. Denne dokumentasjonen baserer seg på kommentarer og Python 'docstrings'[53], men presenterer all denne informasjonen i et mer leslig format, som også personer uten programmeringsbakgrunn kan dra nytte av. Videre så inneholder denne dokumentasjonen informasjon om alle klasser, funksjoner, variabler, call-graphs og filer som det utviklede rammeverket består av.

[6] Bilder levert av dette webgrensesnittet er ikke designet med utgangspunkt i teori rundt ergonomi eller intuitivt design i digitale medier men kun ment som eksempler.

[7] `http://getbootstrap.com/`

[8] `http://designmodo.github.io/Flat-UI/`

[9] http://doxygen.org

5.7 Konklusjon og drøfting av prototype

I dette kapittelet har en fungerende prototype blitt presentert. Den er utviklet etter de rammer og kravspesifikasjon som er gitt tidligere i rapporten, og viser dermed at kravspesifikasjonen fungerer i praksis. Brukerinteraksjon i form av kommandolinjeargumenter er utviklet og viser hvor enkelt systemet er i bruk. Det har blitt utviklet to forskjellige plugins for å vise at integrasjonen mellom rammeverk og plugins fungerer godt. Disse eksemplene er lagt frem for å vise hvor enkelt og oversiktlig utvikling av nye plugins kan være. Ved å bruke prototypen på et datasett utdelt av oppdragsgiver vises det at rammeverk og plugins fungerer i et praktisk og realistisk scenario. Prototypen er utviklet på en oversiktlig og forståelig måte som gjør arbeidet med å bygge videre på implementasjonen enkel. Dette er oppnådd ved å utvikle et modulært system, noe som er et av kravene nevnt i *Kapittel 4 - Kravspesifikasjon og design (s. 18)*. En god og oversiktlig filstruktur er utviklet og dokumentert godt for å gi et verktøy som er enkelt å bruke. God dokumentasjon til prototypen er også generert for å kunne levere en god oversikt over kildekoden.

Det har blitt vist at prototypen fungerer i integrasjon med andre programmer, noe som er sentralt for funksjonaliteten for et slikt system. Denne implementasjonen illustrerer behovet som er kommet frem i det gjennomførte analyse- og kartleggingsarbeidet som er gjort i *Kapittel 3 - Deteksjon og analyse (s. 10)*. Det har ikke blitt gitt noen ytelsesmessige krav så bachelorgruppens implementasjon har ikke fokusert på effektivt ressursbruk og lavt tidsbruk. Prototypen er som nevnt tidligere kun en prototype, og den fungerer dermed godt under kontrollerte omgivelser, men har noen mangler for å kvalifisere seg som et ferdig produkt klar til bruk i et produksjonsmiljø. De mangler som prototypen har er ikke feil som har sneket seg inn under utvikling, men heller mangler som det har blitt tatt bevisste valg om ikke å fokusere på. Hovedfokuset for prototypen har vært å vise at tankegangen presentert tidligere i rapporten fungerer. I *Kapittel 6 - Videre arbeid (s. 49)* presenteres og diskuteres de elementer som burde fikses, videreutvikles og testes før et slikt system settes i produksjon.

6 Videre arbeid

6.1 Analyserbare attributer

6.1.1 Terskler for å vurdere hendelser

Prototypen utviklet for denne rapportens rammeverk inneholder to plugins som beskrevet i *Avsnitt 5.4 - Plugins (s. 42)*. Begge disse pluginene er utviklet hvor resultater gis som en boolsk verdi, med andre ord resultatene fra disse analysene gis enten som en `True` eller `False` ettersom den vurderes som ondsinnet eller ikke i henhold til den respektive analyserbare attributen. I mange situasjoner kan dette være en begrensende og uheldig egenskap. Et godt eksempel her er med vurdering av omdømme for eksempelvis domenenavn og nettsider. Det kan være flere kilder som sier at en side er ondsinnet mens like mange kilder kan gi motstridende beskjed. Overføres dette prinsippet til rammeverket kan det være svært nyttig å kunne ha en skala som representerer hvor ondsinnet eller for eksempel hvor sterk indikasjon man har i en gitt analyse. I *Avsnitt 4.2.3 - Plugins (s. 29)* er det allerede definert et krav til at plugins skal returnere sitt analyseresultat som en desimalverdi mellom 0.0 og 10.0. Dette gir et godt utgangspunkt for å kunne tilby en slik funksjonalitet. For potensielt videre arbeid med dette rammeverket og utvikling av plugins anbefales det sterkt å implementere en ikke-boolsk vurdering av forskjellige analyser, så fremt dette er både logisk og mulig å gjøre.

Et annet viktig tema med tanke på analysen som gjennomføres i de enkelte plugins er den faktiske prosessen av å vurdere om en query er ondsinnet eller ikke. Hvis en utvikler for eksempelvis vil flagge / trigge på alle queries som omhandler toppdomenenivået *.su*[1] så er dette mulig å gjennomføre med en enkel `if-else`-funksjon med et `True/False` resultat. I andre scenarioer vil dette være vanskelig å gjennomføre uten en mengde forskning gjennomført på forhånd. Et godt eksempel her er med antall numeriske tegn i et domenenavn. I denne sammenhengen kommer det opp mange spørsmål som hvor mange tall er mer enn vanlig, hva er vanlig, hvilket datagrunnlag skal man benytte, og skal man se det i sammenheng med TLD, TTL eller noe annet. Det anbefales på det sterkeste at dette området arbeides videre med i både akademia og hos potensielle videreutviklere av rammeverket. Det er få til ingen kilder for slike spesifikke terskler av analyserbare attributter i domenenavn, og nytteverdien til slike terskler er høy.

6.1.2 Nye analyserbare attributer

Denne rapporten har oppsummert en rekke forskjellige analyserbare attributer, disse kan sees i *Tabell 2 - Attributer (s. 14)*. Denne listen er ikke angitt som fullstendig, og det er uten tvil flere aspekter ved datasettet til pDNS-sensorer som kan tilby flere analyserbare attributer. Det anbefales at arbeidet med å videre definere og forske på nåværende analyserbare attributer vedlikeholdes og at man ikke vurderer dagens attributter som uttømmende. Denne prosessen anbefales å bli håndtert som sikkerhetsarbeid generelt: en kontinuerlig prosess. Det vil alltid oppstå nye sårbarheter og skadevareteknikker, og her kan pDNS være en potensielt svært viktig datakilde for etterforskning.

[1] TLD for Sovjetunionen

6.2 Rammeverket

6.2.1 Arkitektur

Dagens arkitektur for rammeverket benytter seg av en løsning hvor det komplette datasettet leses inn i minnet før det sendes videre til analyse. Ved mindre analyser (typisk $\leq 10,000$ pDNS-oppføringer iht *Figur 21 - Kjøring av tidstest (s.41)*) vil ikke dette by på nevneverdige tids- eller ressursmessige utfordringer. Skal derimot rammeverket utsettes for større påkjenninger enn slike datasett vil arkitekturmessige forandringer være nødvendige for å opprettholde en akseptabel analysetid og ressursbruk. Den nedsatte analysetiden i tilfeller hvor størrelsen på datasettene er utfordringen kan anses å stamme fra begrensede ressurser. I disse tilfellene må tregere lagringsenheter benyttes av operativsystemet hvis minneforbruket blir for høyt, noe som vil føre til økt tidsforbruk.

En mulighet for å raskt omforme arkitekturen til å håndtere større datamenger er å analysere hver pDNS-oppføring i det den leses inn fra fil i stedet for at rammeverket skal lese inn hele datasettet før analysen kan begynne. Andre muligheter for dette kan være at rammeverket gjennomfører en rask vurdering av datasettets størrelse og innhold for så vurdere om den skal deles opp i mindre deler for å opprettholde en akseptabel analysetid.

Slik analysen gjennomføres med rammeverkets prototype vil hver plugin aktivert i systemet medføre et resultat skrevet til ønsket destinasjon. Dette vil gi en analytiker som skal vurdere dette systemet en stor mengde data som må korreleres og vurderes etter at en analyse fra rammeverket er gjennomført. Som en løsning på dette kan det vurderes å omstrukturere dagens løsning til at samtlige resultater fra de individuelle pluginsene aggregeres og vurderes / vektes i sammenheng av rammeverket for å gi et mer sammensatt svar til en analytiker. En slik løsning kan potensielt tilbys som en del av rammeverket, enten om rammeverket endres til å analysere hendelser med denne metodikken som standard eller om det tilbys som en valgfri funksjonalitet. En annen løsning for dette er å utvikle et tilleggsprogram som har som oppgave å aggregere og sammenstille resultatene gitt fra rammeverket.

Dagens løsning kjøres som en prosess med en tråd. Som nevnt har det ikke blitt gjennomført større ytelsestesting eller vurdering av ressursbruk ettersom rammeverket leverer resultater innenfor et akseptabelt tidsrom ved analyse av mindre datasett. For å gjøre rammeverket i stand til å utføre analyse av større datamengder raskere kan en potensiell løsning være å omprogrammere dagens arkitektur til en trådbasert løsning. Med dette kan flere oppføringer og eller flere plugins analyseres samtidig.

6.2.2 Optimalisering og forbedret ressursbruk

Utenom en mindre ytelsestest ved bruk av tidtakning gjennomført i *Avsnitt 5.3.2 - Resultater (s. 41)* har det ikke blitt gjennomført noen form for vurdering og analyse av rammeverkets nåværende ressursbruk. Det kan være et område av stor interesse å gjennomføre flere og mer omfattende tester av ressursbruk for å avdekke potensiell ytelsesforbedring. Ved dagens løsning og tiltenkt bruksområde vil dette muligens ikke ha noen stor nytteverdi. Hvis rammeverkets omfang og bruk eskaleres kraftig i forhold til dette vil nytteverdien av en forbedret og optimalisert løsning være stor.

6.2.3 Programvaresikkerhet

Ved all programvare som utvikles må alltid sikkerheten vurderes. Det kan være flere faktorer som utsetter programvaren for risikoer. I denne prototypen kan dette deles inn i to kategorier, risiko som skyldes feilaktig utvikling og risiko som skyldes usikker utvikling. For den første kategorien med feilaktig utvikling kan en bedre kodegjennomgang og periodisk evaluering av moduler eller funksjonalitet som er under utvikling vurderes. Den andre kategorien - usikker utvikling - er et vanskeligere tema å ta for seg. Det er flere punkter i løpet av en kjøring av rammeverket som tar i mot data fra en sluttbruker og skal analysere dette. Dette kan potensielt være grobunn for ukjente sårbarheter eller scenarioer som gir en angriper eller kanskje bare en bruker som benytter rammeverket feilaktig tilgang til oppførsel rammeverket ikke er designet til. Det burde gjennomføres en kodegjennomgang med tanke på sikkerhet for de mest kritiske punktene av rammeverket - f.eks ved innlesning av data eller annen potensiell sårbar funksjonalitet - for å avdekke slike svakheter. Denne rapporten har produsert en prototype som en POC og har dermed unnlat å gjennomføre en sikkerhetsmessig kodegjennomgang. På generiske områder som f.eks konvertering av datatyper eller importering av moduler har best-practice blitt fulgt med tanke på feilhåndtering.

6.3 Plugins

6.3.1 Pluginstruktur

Den nåværende strukturen og kravene satt til plugins i rammeverket er tilpasset det behovet som har blitt sett i dag, men kan i fremtiden møte på flere utfordringer. Som tidligere nevnt kan det være et svært interessant tema å vurdere potensialet til en arkitektur hvor de forskjellige pluginene bærer med seg en gitt vekting og blir vektet opp mot hverandre for å gi en mer sammensatt vurdering av en enkelt pDNS-oppføring. For å kunne gjennomføre dette må de nåværende rammene for en plugin endres. Dette kan gjennomføres ved at hver plugin sin kode definerer en gitt vekt som blir returnert tilbake sammen med analyse resultatet, eventuelt kan en slik vekting gjøres via de nåværende konfigurasjonsfilene. Et tredje alternativ er å opprette en konfigurasjonsfil som gir analytikere en rekke forskjellige sammensatte analyser. En slik sammensatt analyse vil benytte et utvalg av plugins med tilpasset vekting for de respektive analysene. Eksempler her kan være analyser som ser konkret etter FF-kjennetegn eller etter domenenavn med en antydning til å være generert av en algoritme.

6.3.2 Anbefalinger for utvikling

Denne rapportens prototype har utviklet plugins for TLD og TTL som beskrevet i *Avsnitt 5.4 - Plugins (s. 42)*. Det anbefales etter denne rapportens vurdering å utvikle plugins for de resterende tre analyserbare attributtene beskrevet i *Avsnitt 3.2.2 - Analyserte attributer (s. 14)*.

7 Diskusjon

7.1 Kritikk av oppgaven

7.1.1 Todelt fokus

Som nevnt innledningsvis i denne rapporten samt forklart i detalj i løpet av innholdet er fokusområdet til denne rapporten todelt. Det første fokusområdet er kartlegging og presentasjon av analyserbare attributer i pDNS-data som kan brukes i deteksjon av skadevare. Det andre fokusområdet er å identifisere behovene til et rammeverk som skal kunne analysere disse kartlagte attributene. Denne delen inneholder også implementering og vurdering av et proof-of-concept-programvare. Disse to delene er svært forskjellige selv om de bygger på hverandre. Etterhvert som arbeidet med oppgaven har blitt gjennomført har gruppens medlemmer følt at disse oppgavene kunne ha vært delt opp i hver sitt bachelorprosjekt. Dagens situasjon gjør at man ikke kan gå helt inn i verken kartlegging og analysedelen av prosjektet eller kravspesifisering og design av et slikt rammeverk. Ettersom disse to delene bygger på hverandre gir det oppdragsgiver og veiledere et problem hvis dette skulle ha vært to separate oppgaver. Løsninger på dette kunne vært en ny løsning hvor to prosjektgrupper aktivt jobbet sammen, eller om oppdragsgiver hadde en enda tydligere definert oppgavebeskrivelse til hver av oppgavene.

7.1.2 Manglende relevant forskning for attributanalyse

I løpet av arbeidet med å kartlegge samt vurdere forskjellige analyserbare attributter i pDNS-data ble det raskt klart at dette var et fagfelt med svært lite arbeid tilgjengelig for å begrunne våre funn eller vurderinger. Når det skulle vurderes verdien eller troverdigheten til en analyserbar attribut ved en positiv trigger i en alarm oppstod det mange spørsmål. Hva eller hvor er terskelen og skillet mellom ondsinnet og legitimt? Et godt eksempel her er antall numeriske tegn. Svært få av topp besøkte domenenavn inneholder numeriske tegn, så hvor går grensen for at det er nok numeriske tegn til at det mest sannsynlig er et domene med ondsinnede hensikter? Etter mye søking på åpne og lukkede kilder på internett, konsultering med flere akademikere innen informasjonssikkerhet samt personer fra næringslivet har slik data vært vanskelig å finne. Det - antagelse basert på at prosjektgruppen ikke har funnet slik data - mangler forskning og eller offentliggjøring av data som omhandler universielle terskler som kan skille mellom ondsinnet og ikke-ondsinnet domenenavn. Dette er selvfølgelig en svært vanskelig oppgave ettersom man finner store forskjeller i DNS-trafikk i forskjellige land, regioner og organisasjoner. Denne manglende forskningen er noe prosjektets medlemmer sterkt anbefaler en institusjon som for eksempel Norwegian Information Security laboratory (NISlab) å forfølge videre.

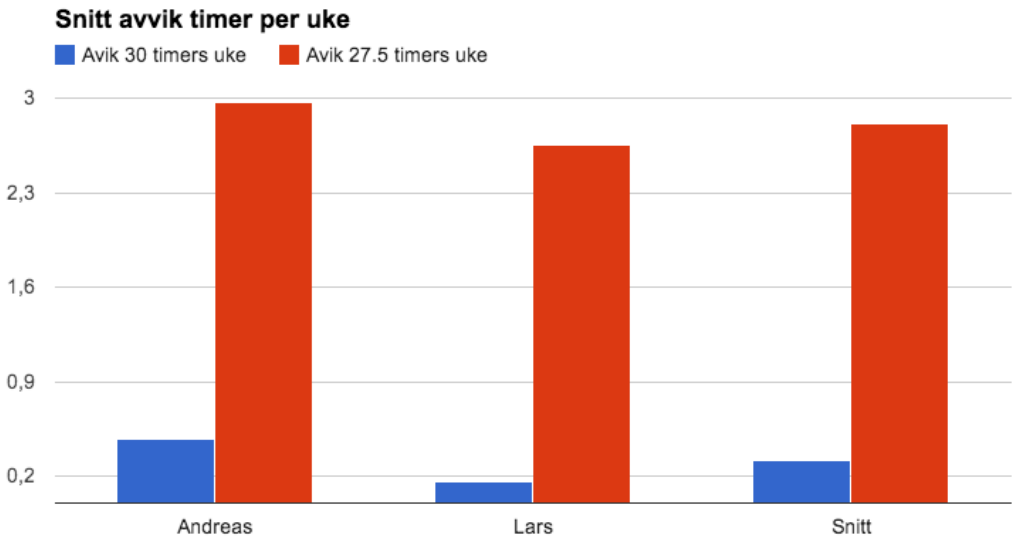
7.2 Evaluering av gruppens arbeid

7.2.1 Arbeidsform og prosjektstyring

Fra starten av prosjektet følte begge medlemmer at alt var tilrettelagt for en best mulig prosjektstyring og gjennomføring av prosjektet. Gruppens medlemmer kjente også hverandres studieteknikker - styrker samt svakheter - fra før av. En scrum-inspirert metodikk for prosjektet kombinert med en flat struktur uten en tildelt leder la grunnlag for et arbeidsmiljø som var svært engasjerende, effektivt og åpent. Samtidig som denne strukturen førte til et effektivt arbeidsmiljø medførte det også til en svært ustabil hverdag. Iløpet av hver to ukers sprint kunne det oppstå endringer eller nye oppdagelser som ville sende neste sprint i en litt annen retning. Det var til tider frustrerende at man ikke konkret kunne redusere antall gjenstående arbeidsoppgaver ettersom nye kontinuerlig tilfalt prosjektets backlog. Dette prosjektet har dermed hatt en veldig dynamisk natur, og når gruppen ser tilbake på prosjektperioden så kommer det godt frem at dette passet gruppens medlemmer godt da begge medlemmer trives godt i dynamiske miljøer.

7.2.2 Disponering av tid

I forprosjektet ble det utarbeidet et Gantt-skjema for å beskrive planlagt disponering av tiden som var satt av til prosjektet. I *Appendix A.6 - Revidert Gantt-skjema (s. 65)* vises det reviderte Gantt-skjeamet som gjenspeiler hvordan tiden faktisk ble distribuert. Det er kun mindre omrokeringer og tilpasninger som har blitt gjort. Oppsummert har gruppens estimerte tidsfordeling truffet bra. I løpet av prosjektets gang har gruppens medlemmer loggført arbeidstimer hver dag.



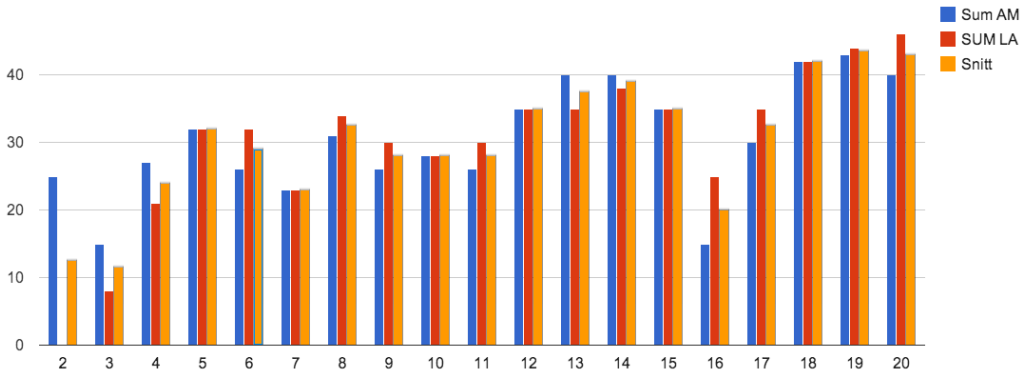
Figur 31: Gjennomsnitt av avvik fra planlagte arbeidstimer per uke

Figur 31 - Gjennomsnitt av avvik fra planlagte arbeidstimer per uke (s.53) viser en oversikt over gjennomsnittlig avvik fra planlagt arbeidstimer per uke, altså i snitt hvor mange timer hver uke har et medlem jobbet mer eller mindre enn det som var planlagt. Denne grafen baserer seg på loggføring gjort individuelt av gruppens medlemmer. Hver uke er summert opp til en total som sammenlignes med en total som baserer seg på en 30 timers uke og en med 27.5 timers uke. I snitt har prosjektets medlemmer arbeidet

$\approx$ 30.31 timer per uke, noe som er rett i overkant av det tidligere avtalte nivået som var ønsket fra hvert medlem.

I prosjektplanen ble det estimert og avtalt at en innsats på rundt 27.5 til 30 timer per uke var nødvendig for å kunne levere et kvalitetsprosjekt. Prosjektet har opplevd litt avvik fra uke til uke, noen uker med 40 timer eller mer arbeidet og noen uker med mindre enn 20 timer. *Figur 32 - Loggførte timer per uke (s.54)* viser en graf over hver uke i prosjektet samt antall registrerte arbeidstimer. Verdiene på Y-aksen er antall timer registrert totalt den uken, X-aksen er uketallet dette ble registrert. Fargene er som følger: Blå = Andreas Moe, Rød = Lars Christian Andersen, Gul = Snitt av registrerte timer den respektive uken.

I forprosjektet ble det estimert et område mellom 1200 ± 50 arbeidstimer for prosjektet. Denne beregningen tar høyde for de resterende ukene etter at rapporten er innlevert, noe som ikke vil bli vedlagt i sluttraporten grunnet innleveringsfrist 19. Mai 2014. Med nye beregninger som kun tar høyde for arbeidstid satt av til sluttraporten (19 uker) vil estimert arbeidsinnstas være 1092.5 ± 47.5. Totalt loggførte timer for samtlige av prosjektets medlemmer er registrert til å være 1152, rett over det estimerte.



Figur 32: Loggførte timer per uke

8 Konklusjon

Sluttresultatet av dette bachelorprosjektet har vært et tredelt resultat. Det har blitt gjennomført en klartlegging, identifisering og overordnet analyse av potensielle analyserbare attributter som finnes i et pDNS-datasett. Dette ble presentert i *Kapittel 3 - Deteksjon og analyse (s. 10)*. Videre arbeidet med analyserbare attributter, og dialog med vår oppdragsgiver mnemonic, lagt grunnlaget for en kravspesifkasjons- og designdokument. Dette ble presentert i *Kapittel 4 - Kravspesifikasjon og design (s. 18)*. Videre ble det presentert en POC av det skisserte rammeverket i *Kapittel 5 - Prototype (s. 33)*.

Tidlig i rapporten ble det som nevnt, identifisert et utvalg av teknikker som skadevare kan benytte seg av for å kommunisere over datanettverk. Dette ble presentert sammen med potensielle attributer som kan brukes til deteksjon og videre analyse av de slik ondsinnet kommunikasjon. Dette arbeidet med identifisering og kartlegging, viser hvor stor den potensielle nytteverdien til pDNS med tanke på deteksjon av skadevare kan være. Tidligere arbeid i form av forskingsartikkler og andre fagliteratur er brukt til å presentere hvordan attributene kan brukes i deteksjon.

En kravspesifikasjon og et designdokument, for et velfungerende rammeverk som tar utgangspunktet i kartleggingen gjort med tanke på analyserbare attributter, er utviklet. Denne kravspesifikasjonen tar for seg behov og ønsker gitt fra oppdragsgiver ved periodisk dialog, samt krav prosjektets medlemmer har avdukket underveis med arbeidet i tidligere kappitler. Denne kravspesifikasjonen samt designdokumentet bygger rammene og kravene for et fullstendig rammeverk som skal kunne analysere både de nåværende kartalgte attributtene, men også fremtidige via et fleksibelt plugin system.

Det har blitt utviklet en fungerende prototype for å presentere at rammeverket er mulig i praksis, en såkalt POC. Denne prototypen ble utviklet i programmeringsspråket Python. Prototypen basserer seg på kravspesifikasjonen presentert, og gir et eksempel til leseren på hvordan et slikt system skal og kan implementeres. Det har blitt utviklet to plugins for å vise at integrasjonen mellom rammeverk og plugins fungerer godt. Dette viser også hvor enkelt det er å utvikle og integrere nye plugins for et slikt rammeverk. Mal for nye plugins er utviklet, og prototypen leveres med en god og fyldig dokumentasjon.

Som et endelig sluttprodukt leveres en innledende kartlegging av teknikker og attributter som kan avdekke disse teknikkene som skadevare benytter seg av. Videre leveres det en åpen prototype som kan bistå fremtidig utvikling av lignende eller inspirerte løsninger for deteksjon av skadevare som tar utgangspunkt i pDNS. Videre er denne oppgaven i sin helhet til å faglig understøtte valgene prosjektets medlemmer har tatt for dette arbeidet.

Kampen mot trusselaktører og deres arsenal av skadevare og teknikker er en kontinuerlig prosess, hvor hvert eneste skritt mot sikrere informasjonssystemer teller. Deling av kunnskap og informasjon er et viktig tema innen informasjonssikkerhet, som ellers er omgitt av mye hemmelighold. Med denne rappoten har vi delt vår kunnskap, våre funn og våre anbefalninger. Med dette vil vi påstå at videre arbeid med analyse av pDNS-data og deteksjon av ondsinnet aktivitet via et rammeverk som rapporten har presentert, er et skritt, mot sikkrere informasjonssystemer.

Bibliografi

[1] Mapping the Mal Web - The World's riskiest domains;. (Besøkt 12. Apr. 2014). http://promos.mcafee.com/en-US/PDF/MTMW_Report.pdf.

[2] Casey E. Investigating Sophisticated Security Breaches. Commun ACM. 2006 Feb;49(2):48–55. Available from: http://doi.acm.org/10.1145/1113034.1113068.

[3] Whitman M, Mattord H. Principles of information security. Cengage Learning; 2011.

[4] Awan I, Blakemore B. Policing Cyber Hate, Cyber Threats and Cyber Terrorism. Ashgate Publishing Company; 2012.

[5] Weimer F. Passive DNS replication. In: FIRST Conference on Computer Security Incident; 2005. .

[6] Bilge L, Kirda E, Kruegel C, Balduzzi M. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In: NDSS; 2011. .

[7] Holz T, Gorecki C, Rieck K, Freiling FC. Measuring and Detecting Fast-Flux Service Networks. In: NDSS; 2008. .

[8] Yadav S, Reddy AKK, Reddy A, Ranjan S. Detecting algorithmically generated malicious domain names. In: Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. ACM; 2010. p. 48–61.

[9] Schwaber K, Beedle M. gilè Software Development with Scrum. 2002;.

[10] Mockapetris P. Domain names - implementation and spesification; 1987. .

[11] Mockapetris P. Domain names - concepts and facilities; 1987. .

[12] Albitz P. DNS and Bind. O'Reilly Media, Inc.; 2001.

[13] Caglayan A, Toothaker M, Drapeau D, Burke D, Eaton G. Real-time detection of fast flux service networks. In: Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology. IEEE; 2009. p. 285–292.

[14] Perdisci R, Corona I, Giacinto G. Early detection of malicious flux networks via large-scale passive DNS traffic analysis. Dependable and Secure Computing, IEEE Transactions on. 2012;9(5):714–726.

[15] Abu Rajab M, Zarfoss J, Monrose F, Terzis A. A multifaceted approach to understanding the botnet phenomenon. In: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement. ACM; 2006. p. 41–52.

[16] Holz T, Gorecki C, Freiling F, Rieck K. Detection and mitigation of fast-flux service networks. In: Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08); 2008. .

[17] Yu S. Malicious Networks for DDoS Attacks. In: Distributed Denial of Service Attack and Defense. Springer; 2014. p. 15–29.

[18] Fitzgibbon N, Wood M. Conficker. C: A technical analysis. SophosLabs, Sophon Inc. 2009;.

[19] Looking Back at Murofet, a ZeuSbot Variant's Active History;. (Besøkt 13. Apr. 2014). https://blog.damballa.com/archives/1008.

[20] Intrusion Detection FAQ: Statistical based approach to Intrusion Detection;. (Besøkt 17. Apr. 2014). http://www.sans.org/security-resources/idfaq/statistic_ids.php.

[21] Di Pietro R, Mancini LV. Intrusion detection systems. Springer; 2008.

[22] Roesch M, et al. Snort: Lightweight Intrusion Detection for Networks. In: LISA. vol. 99; 1999. p. 229–238.

[23] Kruegel C, Toth T. Using decision trees to improve signature-based intrusion detection. In: Recent Advances in Intrusion Detection. Springer; 2003. p. 173–191.

[24] Lupton R. Statistics in theory and practice. Princeton University Press; 1993.

[25] Bilge L, Dumitras T. Before We Knew It. 2012;.

[26] De Laet G, Schauwers G. Network security fundamentals. Cisco Press; 2004.

[27] Passerini E, Paleari R, Martignoni L, Bruschi D. Fluxor: Detecting and monitoring fast-flux service networks. In: Detection of intrusions and malware, and vulnerability assessment. Springer; 2008. p. 186–206.

[28] Perdisci R, Corona I, Dagon D, Lee W. Detecting malicious flux service networks through passive analysis of recursive dns traces. In: Computer Security Applications Conference, 2009. ACSAC'09. Annual. IEEE; 2009. p. 311–320.

[29] Elz R, Bush R. Clarifications to the DNS spesification;. .

[30] Costello p. Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA);. .

[31] Alexa Top 1M websites;. (Besøkt 5. Mar. 2014). http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[32] Moz top 500 domains;. (Besøkt 5. Mar. 2014). http://moz.com/top500/domains/csv.

[33] Domain Name Analysis;. (Besøkt 21. Feb. 2014). http://datagenetics.com/blog/march22012/index.html.

[34] Malware domain list;. (Besøkt 5. Mar. 2014). http://www.malwaredomainlist.com/mdlcsv.php.

[35] Stalmans E, Irwin B. A framework for DNS based detection and mitigation of malware infections on a network. In: Information Security South Africa (ISSA), 2011. IEEE; 2011. p. 1–8.

[36] Yadav S, Reddy AN. Winning with DNS failures: Strategies for faster botnet detection. In: Security and Privacy in Communication Networks. Springer; 2012. p. 446–459.

[37] Dhamija R, Tygar JD, Hearst M. Why phishing works. In: Proceedings of the SIGCHI conference on Human Factors in computing systems. ACM; 2006. p. 581–590.

[38] McGrath DK, Gupta M. Behind Phishing: An Examination of Phisher Modi Operandi. LEET. 2008;8:4.

[39] Ma J, Saul LK, Savage S, Voelker GM. Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2009. p. 1245–1254.

[40] Chong C, Liu D, Lee W. Malicious url detection;.

[41] Antonakakis M, Perdisci R, Nadji Y, Vasiloglou N, Abu-Nimeh S, Lee W, et al. From throw-away traffic to bots: Detecting the rise of dga-based malware. In: Proceedings of the 21st USENIX security symposium; 2012. .

[42] Quick Stats and Visuals on the Likelihood of a Malicious domain;. (Besøkt 12. Apr. 2014). http://labs.umbrella.com/2013/01/08/quick-stats-and-visuals-on-the-likelihood-of-a-malicious-domain/.

[43] Zhou Yl, Li Qs, Miao Q, Yim K. DGA-Based Botnet Detection Using DNS Traffic. Journal of Internet Services and Information Security (JISIS);3(3/4):116–123.

[44] Yarochkin F, Kropotov V, Huang Y, Ni GK, Kuo SY, Chen IY. Investigating DNS traffic anomalies for malicious activities. In: Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on. IEEE; 2013. p. 1–7.

[45] Shafranovich Y. Common Format and MIME Type for Comma-Separated Values (CSV) Files; 2005. .

[46] Van Rossum G, Warsaw B, Coghlan N. Style guide for python code. http://legacypythonorg/dev/peps/pep-0008/. 2001-2014;.

[47] Patel A JEHJSMSM Picard A. Google Python Style Guide;. (Revisjon 2.59 - Besøkt 8. Apr. 2014). http://google-styleguide.googlecode.com/svn/trunk/pyguide.html.

[48] Python 3.4.0;. (Besøkt 9. Apr. 2014). https://www.python.org/download/releases/3.4.0/.

[49] Should I use Python 2 or Python 3 for my development activity?;. (Besøkt 9. Apr. 2014). https://wiki.python.org/moin/Python2orPython3.

[50] TrafikkLysProtokollen (TLP);. (Besøkt 9. Apr. 2014). https://www.nsm.stat.no/Arbeidsomrader/Internettsikkerhet-NorCERT/Internettsikkerhet—NorCERT/NorCERT/Kontakt/Trafikk-Lys-Protokollen—TLP/.

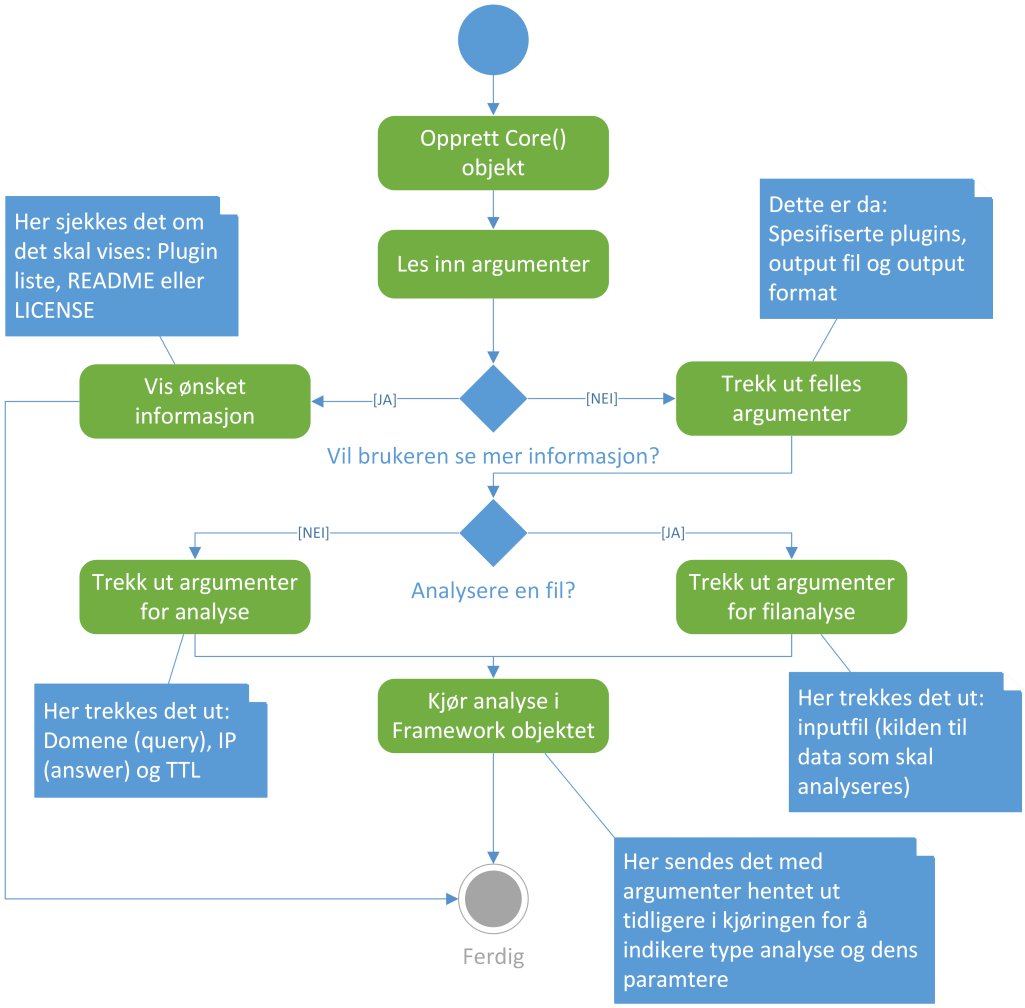
[51] Python Standard Library - Built-in Functions;. (Besøkt 13. Apr. 2014). https://docs.python.org/2/library/functions.html#property.

[52] Flask - Python microframework for rapid web development;. (Besøkt 22. Apr. 2014). http://flask.pocoo.org/.

[53] Van Rossum G, Goodger D. Docstring Conventions. http://legacypythonorg/dev/peps/pep-0257/. 2001-2014;.

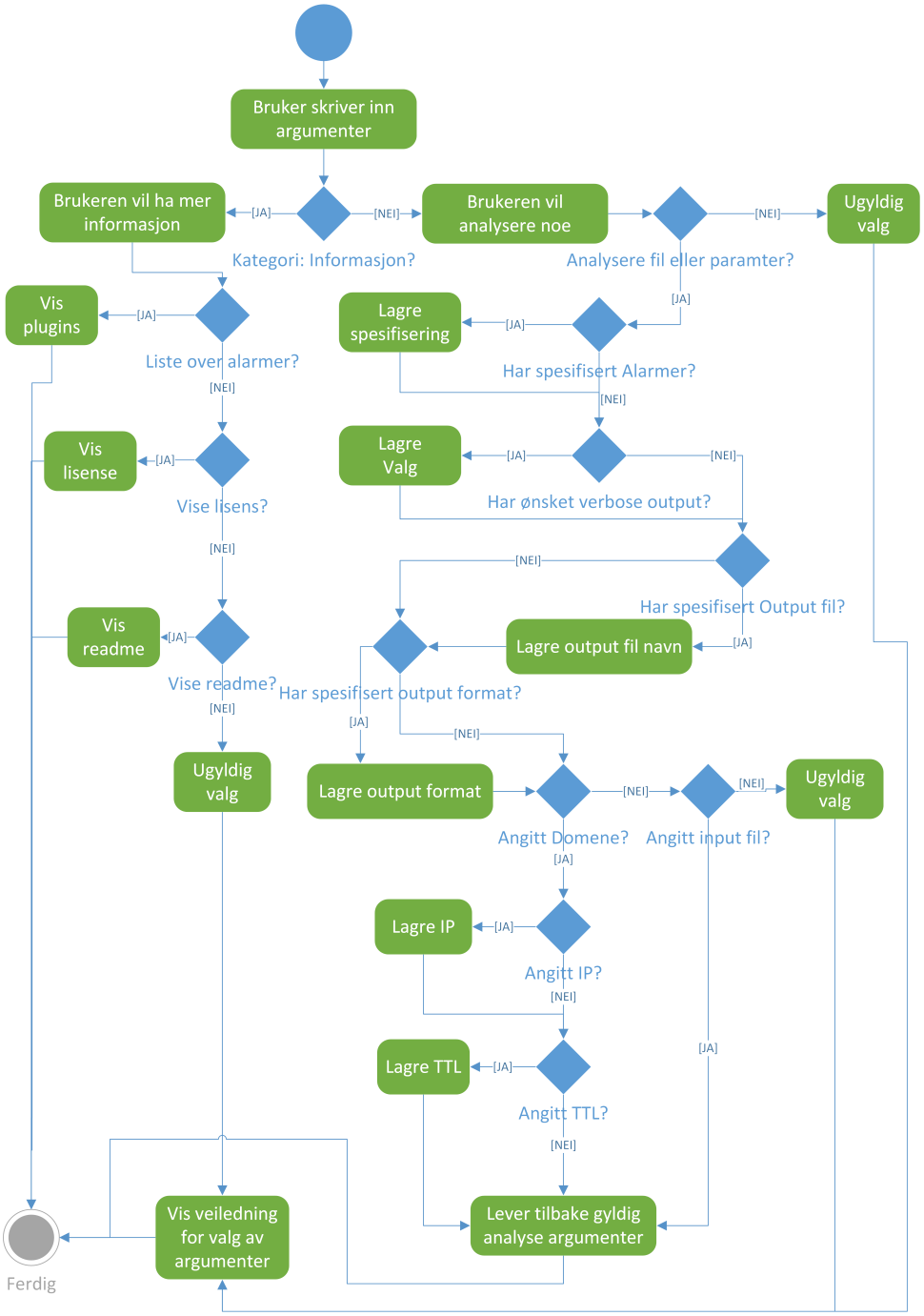
A Figurer og diagrammer

A.1 Kjøring av rammeverket som fil



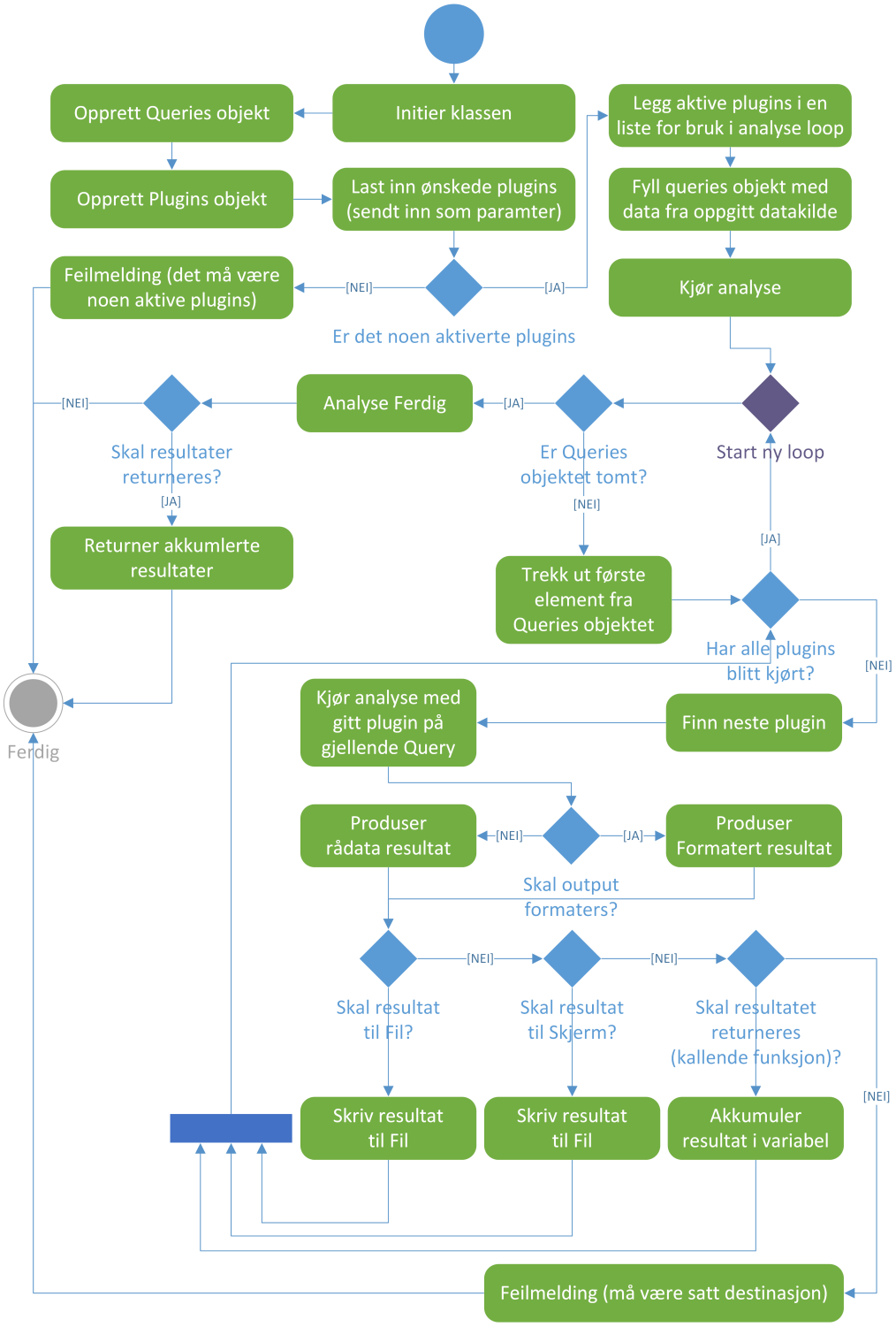
Figur 33: UML-aktivitetsdiagram for å kjøre rammeverket som selvstendig program

A.2 Innlesing av argumenter



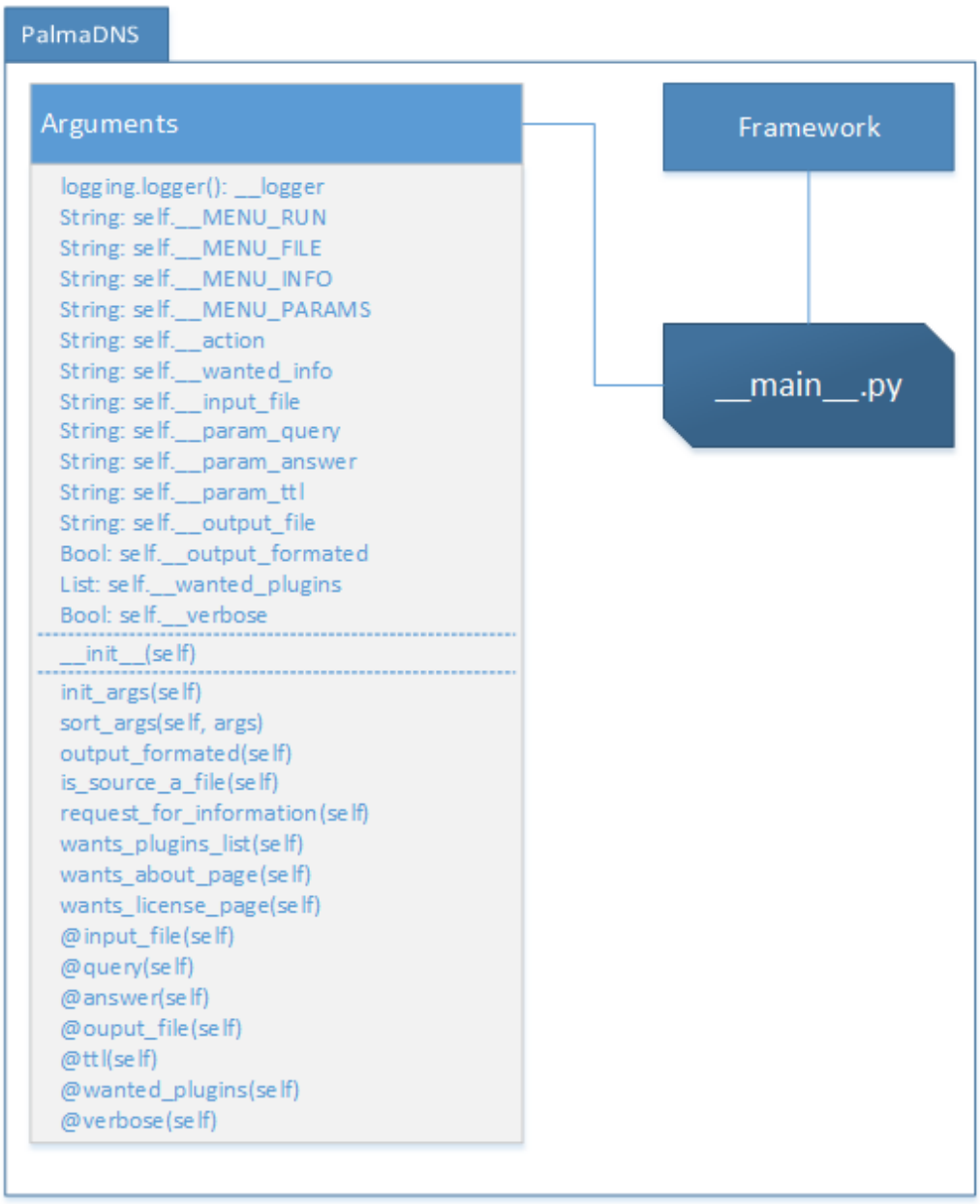
Figur 34: UML-aktivitetsdiagram for validering av kommandolinjeargumenter

A.3 Gjennomføring av analyse



Figur 35: UML-aktivitetsdiagram for å gjennomføre en analyse i rammeverket

A.4 Klassediagram for en Python-implementasjon av PalmaDNS

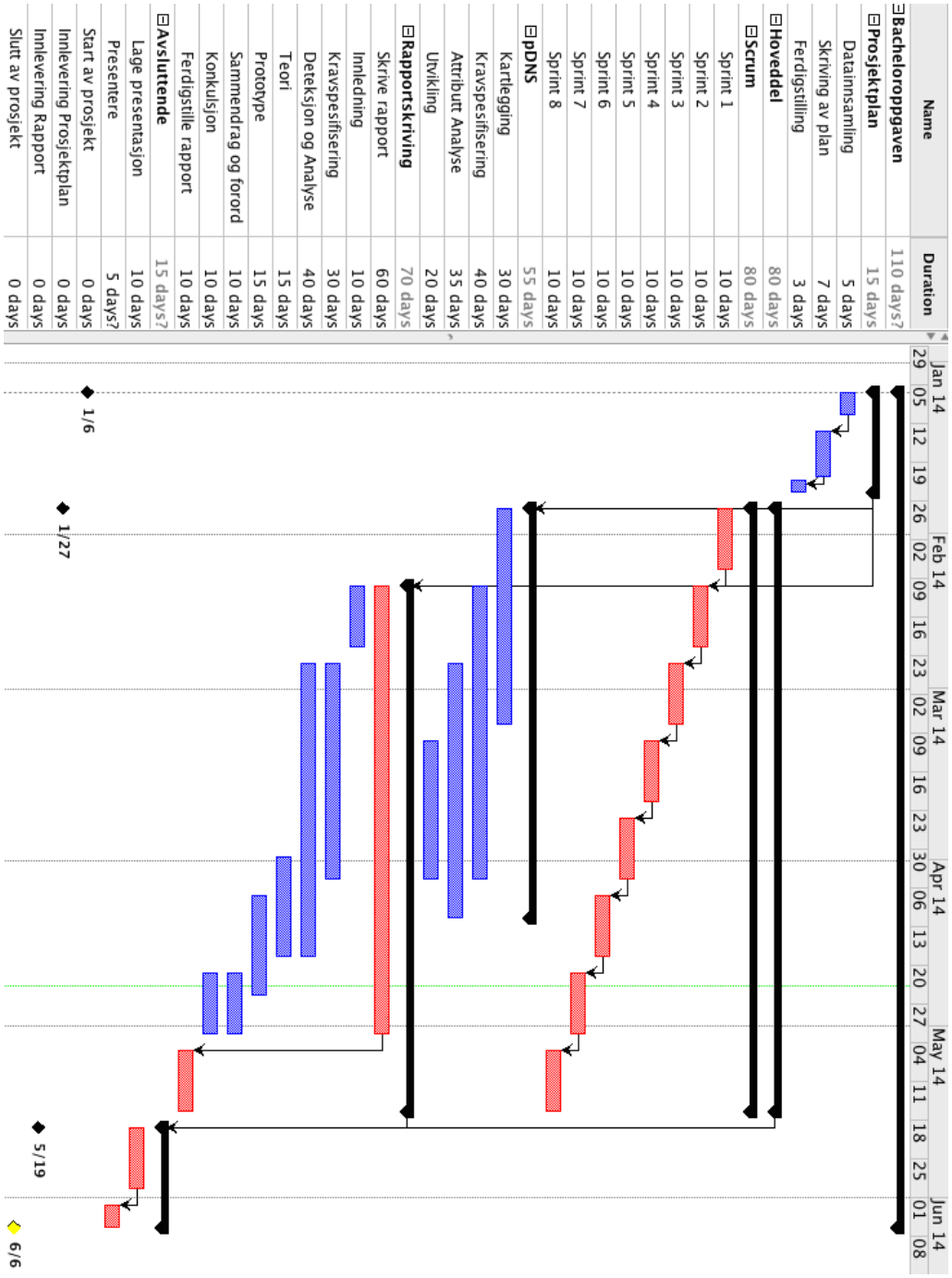


Figur 36: Klassediagram for Palmadns

A.5 Klassediagram for en Python-implementasjon av Framework-komponenter

Figur 37: Klassediagram for Framework

A.6 Revidert Gantt-skjema



Figur 38: Revidert Gantt-skjema

B Kildekode for prototype

B.1 /PalmaDNS/__main__.py

```
"""This is the main file in the package that is PlamaDNS, when this file is run
the main() function is exectued. When this happens potential arguments
are parsed and validated (see framework/arguments.py). A Framework object
is also created (see framework/core.py). After that flow is directed
to where and what the user wanted. This can be:
    - Show information about the framework
    - Analyze a file
    - Analyze commandline arguments (domain, ip, etc given at execution)
"""

__author__ = "Andreas Moe and Lars Christian Andersen"
__copyright__ = "Copyright 2014"
__license__ = "The MIT License (MIT)"
__version__ = "0.1"

try:
    from Framework import core
    from arguments import Arguments
    import logging                              # For logging capabilities
    import logging.config                       # Configfile for logging
    import os
    import ConfigParser
except ImportError, e:
    print 'Could not import module!'
    print 'ERROR - [' + str(e) + '] [' + __name__ + ']'
    raise SystemExit

def main():
    """This is the main() function that is run when this file is executed from
    the shell. This file will set ut the logging configuration, read and
    parse the commandline arguments through the Arguments() class, after
    that this function will give the user the wanted action they were after.
    Either showing the user more information about the program or running
    an analysis.
    """

    path = os.path.dirname(os.path.realpath(__file__))
    config_path = path + '/Framework/config/logging.ini'

    args = Arguments()

    # Get the configuration file that the user wanted
    try:
        config = ConfigParser.ConfigParser()
        config.read(config_path)
        original_level = config.get('logger_root', 'level')

        # Set a new logging level if wanted
        if args.verbose and original_level != 'DEBUG':
            config.set('logger_root', 'level', 'DEBUG')
            with open(config_path, 'wb') as cfgfile:
                config.write(cfgfile)

        logging.config.fileConfig(config_path)
        logger = logging.getLogger(__name__)

    # If a section if the config file was not found
    except ConfigParser.NoSectionError, e:
        print 'ERROR - [' + str(e) + '] [' + __name__ + ']'
        raise SystemExit

    # If an option within a section in the config file was not found
    except ConfigParser.NoOptionError, e:
        print 'ERROR - [' + str(e) + '] [' + __name__ + ']'
        raise SystemExit

    c = core.Core()

```

```
69      # If the user requested more information
70      if args.request_for_information():
71          logger.debug('Directing to information segment of framework')
72          if args.wants_plugins_list():
73              c.show_plugins()
74
75          elif args.wants_about_page():
76              with open(path + '/README.txt', 'r') as f:
77                  content = f.read()
78                  print content
79
80          elif args.wants_license_page():
81              with open(path + '/LICENSE.txt', 'r') as f:
82                  content = f.read()
83                  print content
84
85      # If the user wanted to perform an analysis
86      else:
87          logger.debug('Directing to analysis segment of framework')
88
89          CHOICE_YES = 'Y'
90          CHOICE_NO = 'N'
91
92          # Check if the output file exists and if so ask if the user
93          # wants to override it.
94          if args.output_file != None:
95              if os.path.isfile(args.output_file):
96                  logger.debug('Outputfile allready exsists')
97                  choice = ''
98                  msg = 'Do you want to overwrite %s? (Y/n): ' % args.output_file
99                  while choice != CHOICE_YES and choice != CHOICE_NO:
100                     choice = raw_input(msg)
101                     if choice == '':
102                         choice = CHOICE_YES
103                     else:
104                         choice = choice.upper()
105
106                 if choice == CHOICE_YES:
107                     os.remove(args.output_file)
108                     logger.debug('User wants to overwrite output file')
109                 else:
110                     logger.info('Did not want to overwrite file, quitting')
111                     raise SystemExit
112
113         # If analysis is to be done from data on a file or if not
114         # then run analysis on the given paramters in the commandline arguments
115         c.set_want_output_formated(args.output_formated())
116         c.set_wanted_plugins(args.wanted_plugins)
117
118         if args.output_file:
119             c.set_output_destination(core.DEST_FILE)
120             c.set_ouput_file(args.output_file)
121         else:
122             c.set_output_destination(core.DEST_SCREEN)
123
124         if args.is_source_a_file():
125             c.import_file(args.input_file)
126         else:
127             c.import_single(args.query, args.answer, args.ttl)
128
129         c.run()
130
131         # Revert back to the original logging level
132         if args.verbose and original_level != 'DEBUG':
133             config.set('logger_root', 'level', original_level)
134             with open(config_path, 'wb') as cfgfile:
135                 config.write(cfgfile)
136
137 if __name__ == '__main__':
138     try:
139         main()
140     except SystemExit, e:
141         print 'ERROR - PalmaDNS experienced an error and was exited'
142         print 'Exiting...'
```

B.2 /PalmaDNS/__init__.py

```
"""The PalmaDNS root folder is a container for the PalmaDNS package and for
the CLI capabilities given through __main__.py and arguments.py.
"""

__all__ = ['Framework']
import Framework
```

B.3 /PalmaDNS/arguments.py

```
"""The arguments.py file is the module that holds the Arguments class which
does all the work with initializing and gathering commandline arguments
from the user. There is no other function to this file than to include the
needed modules, and to hold the Arguments class. See the arguments
class docs for more information.
@package PalmaDNS
@filename arguments.py
@author Andreas Moe & Lars C. Andersen
"""

try:
    import logging
    import argparse
except ImportError, e:
    print 'Could not import module!'
    print 'ERROR - [' + str(e) + '] [' + __name__ + ']'
    raise SystemExit

class Arguments:
    """This is the class that delivers all the arguments handeling and parsing
    to this framework.

    This class does mainly three things:
    1: Initiates an arguments parser and defines the different arguments that
    can be set, given or that can / cannot co-exist.
    This is done in the self.init_args() method

    2: Sortes through the arguments and sets the recived data into a data-
    structure (self.__data) so that is is easily accessable with
    differenct methods for the framework to decide what the user wanted
    This is done in the self.sort_args() method

    3: Storing and giving out data about the arguments the user gave as
    commandline arguments. The results of part 2 above are stored
    in a common datastructure (self.__data) and this class contains
    a serifes of checking methods (eg. check if fancy formatting)
    is requested. And some property methods (eg. instance.input_file),
    so that theese datas are avaiable for the framework
    """

    # The differnt "sub menus" of the framework
    __MENU_RUN = 'run'
    __MENU_INFO = 'information'
    __MENU_FILE = 'file'
    __MENU_PARAMS = 'params'

    # Types of information a user can request about the framework
    __LIST = 'list'
    __ABOUT = 'about'
    __LICENSE = 'license'

    def __init__(self):
        """Initiates all private variables and runs initialization and
        sorting of commandline arguments.

        Sets all the private variables with regard to the functionality
        the user wanted to 'None'. A logger is created to handle error
        and debugging messages. The args are initialized and sorted at the
        end of the constructor.
        """
        # Collected arguments data
        self.__action = None
        self.__wanted_info = None
        self.__input_file = None
        self.__param_query = None
        self.__param_answer = None
        self.__param_ttl = None
        self.__output_file = None
```

```
69          self.__output_format = None
70          self.__wanted_plugins = None
71          self.__verbose = None
72
73          # Logger for givving debuging information
74          self.__logger = logging.getLogger(__name__)
75
76          # Set the arguments parser(s) data and then collect and parse them.
77          args = self.init_args()
78          self.sort_args(args)
79
80      def is_source_a_file(self):
81          """Checks if the datasource is a file.
82          Checks to see if the source of the data that is to be
83          analyzed is given from a file and returns the answer as
84          a boolean True/False.
85          """
86          if self.__action == self.__MENU_FILE: return True
87          else: return False
88
89      def request_for_information(self):
90          """Checks to see if the user wanted to see more information about
91          this program and returns the answer as a boolean True/False.
92          """
93          if self.__action == self.__MENU_INFO: return True
94          else: return False
95
96      def wants_plugins_list(self):
97          """Checks if the user wanted to see a list of all the plugins that
98          have been added to the 'plugins' directory and returns the
99          answer as a boolean True/False.
100         """
101         if self.__wanted_info == self.__LIST: return True
102         else: return False
103
104     def wants_about_page(self):
105         """Checks to see if the user wanted to be shown the 'about' page
106         for this program (which is the README.txt file) and returns
107         the answer as a boolean True/False.
108         """
109         if self.__wanted_info == self.__ABOUT: return True
110         else: return False
111
112     def wants_license_page(self):
113         """Checks to see if the user wanted to show the license information
114         for this program (which is the LICENSE.txt file) and returns
115         the answer as a boolean True/False.
116         """
117         if self.__wanted_info == self.__LICENSE: return True
118         else: return False
119
120     def output_formated(self):
121         """Returns the value of wether or not the ouput from the future
122         analysis should be given as a raw data format or a stylized
123         format that is more human readable.
124         """
125         return self.__output_format
126
127     @property
128     def verbose(self):
129         """Returns if the user wants verbose output while running the program.
130         This is used for when users want more information about the running
131         of the program and debugging purposes. This will activate all
132         logging events at the level 'DEBUG'.
133         """
134         return self.__verbose
135
136     @property
137     def input_file(self):
138         """Returns what input file the user has given to the program. If
139         the program is going to analyze input parameters this will by
140         default be set to null due to the constructor.
141         """
142         return self.__input_file
143
144     @property
145     def query(self):
146         """Returns the 'query' value that the user gave to the program as a
147         commandline argument if the user is about to analyze data that
148         is not from a file.
```

```
149             """
150             return self.__param_query
151
152         @property
153         def answer(self):
154             """Returns the 'answer' value that the user gave to the program as a
155             commandline argument if the user is about to analyze data that
156             is not from a file.
157             """
158             return self.__param_answer
159
160         @property
161         def ttl(self):
162             """Returns the 'Time-To-Live' value that the user gave to
163             the program as a commandline argument if the user is about
164             to analyze data that is not from a file.
165             """
166             return self.__param_ttl
167
168         @property
169         def wanted_plugins(self):
170             """Returnes the specified plugins that the user wants to
171             run with this program.
172             """
173             return self.__wanted_plugins
174
175         @property
176         def output_file(self):
177             """Returns the output file that the user (potentialy) has given to
178             indicate where to program should write its results.
179             """
180             return self.__output_file
181
182         def init_args(self):
183             """This method sets up the arguments parsers for the different parts
184             of functionality the the framework can deliver. The different parsers /
185             functionality is:
186
187             Parser: The main parser.
188             parser_analyze: For the arguments that can be given with regards to
189             analyzing data.
190
191             parser_info: For the arguments for displaying more information
192             about the framework.
193
194             parser_analyze_file: The parser for arguments regarding
195             file analysis.
196
197             parser_analyze_input: The parser for arguments regarding
198             analysis of commandline input / arguments
199
200
201             Getting more information:   ./palma.py information ...
202             Analyzing something:        ./palma.py run ...
203             Analyzing a file:           ./palma.py run file ...
204             Analysing paramters:        ./palma.py run params ...
205             """
206
207             self.__logger.debug('Initiating arguments')
208             parser = argparse.ArgumentParser(description='PalmaDNS.')
209             subparsers = parser.add_subparsers(dest='cmd')
210
211             # The subparsers (one for analysis and one for information)
212             p_analyze = subparsers.add_parser(self.__MENU_RUN, help='Run analysis')
213             p_info = subparsers.add_parser(self.__MENU_INFO, help='Information')
214             p_analyze_subparsers = p_analyze.add_subparsers(dest='cmd')
215
216             # The two subparsers in the "Run" section of the parser
217             p_analyze_file = p_analyze_subparsers.add_parser(
218                 self.__MENU_FILE,
219                 help='Run analysis on a file'
220             )
221             p_analyze_input = p_analyze_subparsers.add_parser(
222                 self.__MENU_PARAMS,
223                 help='Run analysis on commandline arguments'
224             )
225
226             # Arguments for when analyzing a file
227             p_analyze_file.add_argument(
228                 '-i',
```

```
229             '--input',
230             help='Input file',
231             required=True
232         )
233         p_analyze_file.add_argument(
234             '-p',
235             '--plugins',
236             help='Wanted plugin(s)',
237             metavar='N',
238             nargs='+'
239         )
240 
241         p_analyze_file.add_argument('-o', '--output', help='Output file')
242         p_analyze_file.add_argument(
243             '-f',
244             '--formated',
245             help='Shows fancy output to screen',
246             action='store_true'
247         )
248         p_analyze_file.add_argument(
249             '-v',
250             '--verbose',
251             help='Shows verbose debuging information',
252             action='store_true'
253         )
254 
255         # Arguments for when analyzing commandline parameters
256         p_analyze_input.add_argument('-q',
257             '--query',
258             help='DNS Query',
259             required=True
260         )
261         p_analyze_input.add_argument('-a', '--answer', help='Query answer')
262         p_analyze_input.add_argument('-t', '--ttl', help='Time To Live')
263         p_analyze_input.add_argument(
264             '-p',
265             '--plugins',
266             help='Wanted plugin(s)',
267             metavar='N',
268             nargs='+'
269         )
270 
271         p_analyze_input.add_argument('-o', '--output', help='Output file')
272         p_analyze_input.add_argument(
273             '-f',
274             '--formated',
275             help='Shows fancy output to screen',
276             action='store_true'
277         )
278         p_analyze_input.add_argument(
279             '-v',
280             '--verbose',
281             help='Shows verbose debuging information',
282             action='store_true'
283         )
284 
285         # Informational arguments in the "Info" subparser
286         info_group = p_info.add_mutually_exclusive_group(required=True)
287         info_group.add_argument(
288             '--list',
289             help='List all alarms',
290             action='store_true'
291         )
292         info_group.add_argument(
293             '--about',
294             help='About this program',
295             action='store_true'
296         )
297         info_group.add_argument(
298             '--license',
299             help='The license for this program',
300             action='store_true'
301         )
302 
303         self.__logger.debug('Finished initializing arguments')
304         return parser.parse_args()
305 
306     def sort_args(self, args):
307         """This method is for sorting through the given arguemnts and put
308         them into the common datastructure so that they are easly accessable
```

```
        for the data methods in this class so that the main framework can
        see what the user wanted to be done.

        @param args The arguments object containing the commandline arguments
        given by the user at execution time.
        """

        self.__logger.debug('Started sorting through arguments')
        # Get the information that is commen for both file and params analysis
        if args.cmd == self.__MENU_FILE or args.cmd == self.__MENU_PARAMS:
            self.__output_file = args.output
            self.__wanted_plugins = args.plugins
            self.__output_format = args.formated
            self.__verbose = args.verbose

        # Get the entered information about analysis of a file
        if args.cmd == self.__MENU_FILE:
            self.__action = self.__MENU_FILE
            self.__input_file = args.input

        # Get the enterd information about analysis of commandline arguments
        elif args.cmd == self.__MENU_PARAMS:
            self.__action = self.__MENU_PARAMS
            self.__param_query = args.query

            self.__param_answer = args.answer
            self.__param_ttl = args.ttl

        # Get the wanted type of information about the program
        elif args.cmd == self.__MENU_INFO:
            self.__action = self.__MENU_INFO

            if args.license == True: self.__wanted_info = self.__LICENSE
            elif args.list == True: self.__wanted_info = self.__LIST
            elif args.about == True: self.__wanted_info = self.__ABOUT

        self.__logger.debug('Finnished sorting through arguments')
```

B.4 /PalmaDNS/Framework/core.py

```
"""This file contains the Core() module that binds the entire framework
together. The role if this Core module is to handle the queries that are to
be analyzed into a Queries() instance sort out the wanted destination
formats and types, then run the wanted analysis. Also this file contains some
global variables so that other calling functions / methods can utilize the
same constant naming of destination types and so on (eg. DEST_FILE)
"""

try:
    import os                               # For finding the path to the plugins
    from plugins import Plugins             # For handling the plugins
    from queries import Queries             # Wrapper for a Deque of Querys
    import logging
except ImportError, e:
    print 'Could not import module!'
    print 'ERROR - [' + str(e) + '] [' + __name__ + ']'
    raise SystemExit

SOURCE_FILE = 'file'
SOURCE_PARAMS = 'params'

DEST_FILE = 'file'
DEST_SCREEN = 'screen'
DEST_RETURN = 'return'

class Core:
    """This is the framework class in core.py. The role of this class is to be
    the glue between all the components of this framework. This class will
    hold and administrate the Queries, Plugins and Arguments parseing.
    Do the file / screen writing of results and messages to the user. This
    class will also do the analysis by emptying the Queries deque, and then
    runing that query object into each activaed plugin, and divert the results
    to the wanted destination (file or screen) and in the wanted format.
    """

    def __init__(self):
        """ Initiates the object by firstly creating a logger to handle
        potential debugging and error messages. Afther this a Queries() object
        is created to handle all the individual queries (that will be imported
        at a later time). Then a Plugins() object is created and a path
```

```
41          the the plugins folder is suplied. After that some common variables
42          about the input / ouput configuration are created and set to None.
43          """
44          self.__logger = logging.getLogger(__name__)
45          path = os.path.dirname(os.path.realpath(__file__))
46          self.__queries = Queries()                          # For holding queries
47          self.__plugins = Plugins('%s/plugins' % path)   # For managing plugins
48
49          self.__output_dest = None
50          self.__wanted_plugins = None
51          self.__output_format = False
52          self.__output_file = None
53
54      def set_output_destination(self, destination):
55          """Sets the wanted output destination (file, screen, return).
56          @param destination The destination for the output of this analysis.
57          """
58
59          self.__output_dest = destination
60
61      def set_wanted_plugins(self, plugins):
62          """Sets the wanted plugins that should be used in an analysis.
63          @param plugins The plugins the user wants to use for this analysis.
64          """
65
66          self.__wanted_plugins = plugins
67
68      def set_want_output_formated(self, output_format):
69          """Sets the wanted output format.
70          @param output_format Sets the type of output format the user wants.
71          """
72
73          self.__output_format = output_format
74
75      def set_ouput_file(self, file):
76          """Sets the output file is the destination is going to be a file.
77          @param file Sets the destination file for the analysis results
78          """
79
80          self.__output_file = file
81
82      def import_single(self, query, answer, ttl):
83          """This method is used to import a single pdns query
84          @param query The domain name that was queried
85          @param answer What the query resolved to
86          @param ttl The Time-To-Live for this resolve
87          """
88
89          self.__logger.debug('Importing single query')
90          self.__queries.create_from_params(query, answer, ttl)
91
92      def import_file(self, input_file):
93          """This method is used to import a pdns log file
94          @param input_file The file that is going to be imported.
95          """
96
97          self.__logger.debug('Importing file (%s)' % input_file)
98          self.__queries.create_from_file(input_file)
99
100     def run(self):
101         """This is the function that initializes the framework to run an
102         analysis. The plugins are activated (if any are specified). The type
103         of analysis output if figured out, and the formatting function is
104         also chosen. After all that the desired analysis with its given
105         output destination is run. Happy analysis!
106         """
107
108         # Activate the wanted (or all if none specified) plugins
109         if self.__wanted_plugins:
110             msg = 'Activating selected plugins (%s)' % self.__wanted_plugins
111             self.__logger.debug(msg)
112             for plugin in self.__wanted_plugins:
113                 self.__plugins.activate(plugin)
114         else:
115             self.__logger.debug('Activating all plugins')
116             self.__plugins.activate_all()
117
118         plugins = self.__plugins.activated
119         if len(plugins) == 0:
120             self.__logger.error('No plugins were detected as activated')
```

```
121             raise SystemExit
122
123         if self.__output_format:
124             self.__logger.debug('Fancy output wanted and set')
125             format_func = self.__make_result_formated
126             header_func = self.__header_formated
127         else:
128             self.__logger.debug('Non-fancy output wanted and set')
129             format_func = self.__make_result_rawdata
130             header_func = self.__header_raw
131
132
133         if self.__output_dest == DEST_FILE:
134             analysis_func = self.__analysis_to_file
135             if self.__output_file == None:
136                 msg = "Destination set to file, but no output file set"
137                 self.__logger.error(msg)
138                 raise SystemExit
139
140         elif self.__output_dest == DEST_SCREEN:
141             analysis_func = self.__analysis_to_screen
142         elif self.__output_dest == DEST_RETURN:
143             analysis_func = self.__analysis_to_return
144         else:
145             self.__logger.error('No destination type was set')
146             raise SystemExit
147
148         return analysis_func(plugins, format_func, header_func, self.__output_file)
149
150     def __analysis_to_return(self, plugins, format_fun, header_fun, dummy=None):
151         """
152         Runs through the queries deque and runs all plugins on given
153         query, then formats accoring to format_func choice and
154         writes the results to screen.
155         """
156
157         self.__logger.debug('Starting analysis with output to screen')
158
159         results = {}
160         results['header'] = header_fun()
161         results['data'] = []
162         while not self.__queries.empty:
163             q = self.__queries.pop()
164             for plugin in plugins:
165                 result = plugin.plugin_object.analyze(q)
166                 results['data'].append(format_fun(result, plugin.name, q))
167
168         self.__logger.debug('Finnished analysis with output to screen')
169         return results
170
171     def __analysis_to_screen(self, plugins, format_fun, header_fun, dummy=None):
172         """
173         Runs through the queries deque and runs all plugins on given
174         query, then formats accoring to format_func choice and
175         writes the results to screen.
176         @retval results Always 'None' since output goes to the screen.
177         """
178
179         self.__logger.debug('Starting analysis with output to screen')
180
181         print header_fun()
182         while not self.__queries.empty:
183             q = self.__queries.pop()
184             for plugin in plugins:
185                 result = plugin.plugin_object.analyze(q)
186                 print format_fun(result, plugin.name, q)
187
188         self.__logger.debug('Finnished analysis with output to screen')
189         return None
190
191     def __analysis_to_file(self, plugins, format_fun, header_fun, ofile):
192         """
193         Runs through the queries deque and runs all plugins on given
194         query, then formats accoring to format_func choice and
195         writes the results to file.
196         @retval results Always 'None' since output goes to a file.
197         """
198
199         self.__logger.debug('Staring analysis with output to %s' % ofile)
200
```

```
201             with open(ofile, 'a') as file:
202                 file.write(header_fun() + '\n')
203                 while not self.__queries.empty:
204                     q = self.__queries.pop()
205                     for plugin in plugins:
206                         result = plugin.plugin_object.analyze(q)
207                         alarm = format_fun(result, plugin.name, q)
208                         file.write('%s\n' % alarm)
209 
210             self.__logger.debug('Finnished analysis with output to %s' % ofile)
211             return None
212 
213         def __extract_result_info(self, result, alarm, qobj):
214             """Exctracts the wanted information needed for generating an alarm.
215             @param result The result of a plugins analysis function.
216             @param alarm The name of the plugin that has been run.
217             @param qobj The query object that was analzed (See class Query()).
218             @retval result A list containing all the data.
219             """
220 
221             dns_client = qobj.dns_client.value
222             answer = qobj.answer.value
223             query = qobj.query.value
224 
225             if dns_client == None: dns_client = '-'
226             if answer == None: answer = '-'
227             if query == None: query = '-'
228 
229             return [dns_client, answer, query, alarm, str(result)]
230 
231         def __make_result_formated(self, result, alarm, qobj):
232             """Creates a formated and 'fancy' printout of each alarm result
233             @param result The result of a plugins analysis function.
234             @param alarm The name of the plugin that has been run.
235             @param qobj The query object that was analzed (See class Query()).
236             @retval result A formated string with the reults of this analysis.
237             """
238 
239             data = self.__extract_result_info(result, alarm, qobj)
240             temp = ''
241             temp += data[0].ljust(20) + '| '
242             temp += data[1].ljust(50) + '| '
243             temp += data[2].ljust(50) + '| '
244             temp += data[3] + ' | '
245             temp += data[4]
246             return temp
247 
248         def __make_result_rawdata(self, result, alarm, qobj):
249             """Extracts information about this alarm and joins to a CSV format
250             @param result The result of a plugins analysis function.
251             @param alarm The name of the plugin that has been run.
252             @param qobj The query object that was analzed (See class Query()).
253             @retval result A CSV formated string.
254             """
255 
256             data = self.__extract_result_info(result, alarm, qobj)
257             return ','.join(data)
258 
259         def __header_raw(self):
260             """Creates a header string for when outout is rawdata results
261             @retval header A CSV foramted header string.
262             """
263 
264             header = 'DNS_Client,Answer,Query,Plugin,Score'
265             return header
266 
267         def __header_formated(self):
268             """Creates a header string for when output is formated
269             @retval header A formated header string.
270             """
271 
272             header = '\n'
273             header += 'DNS Client'.ljust(20) + '|'
274             header += ' Answer'.ljust(51) +  '|'
275             header += ' Query'.ljust(51) + '|'
276             header += ' Plugin'.ljust(9) + '|'
277             header += ' Score'
278             header += '\n'
279             header += '-' * 142
280             return header
```

```
281
282     def show_plugins(self):
283         """Shows a formated printout (screen) of all the available plugins.
284         """
285
286         print ''
287         print '-' * 50
288         for plugin in self.__plugins.all_plugins:
289             print 'Name:\t\t%s' % plugin.name
290             print 'Description:\t%s' % plugin.description
291             print ''
```

B.5 /PalmaDNS/Framework/queries.py

```
1   """File for holding the Queries() class, doesnt really do much more than that.
2   See the Queries docstring for more information about what it does.
3   """
4
5   try:
6       import collections          # Deque (double ended queue)
7       import logging              # For logging events
8       from query import Query     # Query object holders
9   except ImportError, e:
10      print 'Could not import module!'
11      print 'ERROR - [' + str(e) + '] [' + __name__ + ']'
12      raise SystemExit
13
14  class Queries:
15      """This is the class for holding all the Passive DNS detected queries that
16      the framework is to anlyze. The concept here is to have a common interface
17      to creating, storing and retriving information about the queries.
18
19      This class will read single queries or read them from file. The data is
20      then stored in a Query object (see /framework/query.py) and that obejct
21      is added to a double ended queue (collections.deque). This is done
22      so that the first element to be added can be analyzed first and that
23      this process should go fast (lists would be slower, etc).
24      """
25
26      def __init__(self):
27          """Initializes the object by creating a logger and a double ended
28          queue for holding individual query objects.
29          """
30          self.__logger = logging.getLogger(__name__)
31          self.__queries = collections.deque()
32
33      @property
34      def empty(self):
35          """This method is just a simple check to se if there is any data in
36          the deque __data. This is used when looping through all the elements
37          or just simply to check the current status. This is configured as
38          property so it can be called by: object.is_empty (as a variable)
39          @retval __queries Wether or not there are any queries in this object.
40          """
41
42          if self.__queries: return False
43          else: return True
44
45      def create_from_params(self, query, answer, ttl):
46          """This method puts the given parameters (domain, ip, ttl) and puts
47          them into a query object that is appended to the deque __data.
48          @param query The domain name that was queried
49          @param answer What the query resolved to
50          @param ttl The Time-To-Live for this resolve
51          """
52
53          temp = Query()
54          temp.answer.value = answer
55          temp.query.value = query
56          temp.ttl.value = ttl
57          self.__queries.append(temp)
58          self.__logger.debug('Created Query object for single query')
59
60      def create_from_file(self, filename):
61          """This method reads the contents of a CSV file that contains passive
62          dns captured data. Each line is read with the help of the csv.reader
63          method. Each line is then put into a query object and the appended
64          to the deque __data.
65
66          The format for this CSV file must be (header):
```

```
67          timestamp,dns_client,dns_server,rr_class,query,query_type,answer,ttl
68
69          timestamp = The timestamp of the event (epoch)
70          dns_client = The client that performed this query
71          dns_server = The server that answer this query
72          rr_class = The class of the Resource Record
73          query = The data (normaly domain) that was queried about
74          query_typ = The type of query response that was given (A, PRT, etc)
75          answer = What did the query resolve to
76          ttl = The time to live of this query response
77
78          @param filename The name for the file that this function reads its
79          passive dns data from.
80          """
81
82          self.__logger.debug('Started creating Query objects from %s' % filename)
83          with open(filename, 'rb') as file:
84              for line in file:
85                  l = line.split('||')
86                  try:
87                      if len(l) != 8:
88                          msg = 'Number of CSV fields not matching wanted format'
89                          raise AttributeError(msg)
90
91                      l[-1] = l[-1].replace('\n', '')
92
93                      # Extract the datafields into comman variables
94                      temp = Query()
95                      temp.timestamp.value = l[0]
96                      temp.dns_client.value = l[1]
97                      temp.dns_server.value = l[2]
98                      temp.rr_class.value = l[3]
99                      temp.query.value = l[4]
100                     temp.query_type.value = l[5]
101                     temp.answer.value = l[6]
102                     temp.ttl.value = l[7]
103                     self.__queries.append(temp)
104
105                 except AttributeError, e:
106                     self.__logger.error(e)
107                     raise SystemExit
108         self.__logger.debug('Finnished creating objects from %s' % filename)
109
110     def pop(self):
111         """This method gets the first element (to the left) with a .popleft()
112         and returns it to the calling function / user
113         @retval query The first query in the double ended queue.
114         """
115         try:
116             query = self.__queries.popleft()
117         except IndexError:
118             self.__logger.warning('Tried to pop data from empty deque')
119             query = None
120
121         return query
```

B.6 /PalmaDNS/Framework/query.py

```
1   """Holds the Query() class, see the Query docstring for more information.
2   """
3
4   class Query:
5       """This is the class that gives the framework objects to be able to store
6       in the queries class. Each instance of this object contains all the
7       possible data that a passive DNS query capture can contain (according
8       the the specifications of this framework). Theese elements are
9
10      timestamp = The timestamp of the event (epoch)
11      dns_client = The client that performed this query
12      dns_server = The server that answer this query
13      rr_class = The class of the Resource Record
14      query = The data (normaly domain) that was queried about
15      query_typ = The type of query response that was given (A, PRT, etc)
16      answer = What did the query resolve to
17      ttl = The time to live of this query response
18      """
19
20      class QueryData(object):
21          """This is a nested class to reduce the amount of code that had to
22          be written. The functionality is the same for all the variables
```

```
        so this class was created.
        """
        def __init__(self):
            """Create the data holder and set it default to None
            """
            self.__data = None

        @property
        def value(self):
            """Returns the contents of the data holder to the caller
            @retval __data The data contained in this object
            """
            return self.__data

        @value.setter
        def value(self, data):
            """Sets the data holder to holde a new value
            @param data The new value for this data holder variable
            """
            self.__data = data

    def __init__(self):
        """Initiates this object by creating a QueryData() variable
        for each of the different attributes that this Query will
        contain.
        """
        self.timestamp = self.QueryData()
        self.dns_client = self.QueryData()
        self.dns_server = self.QueryData()
        self.rr_class = self.QueryData()
        self.query = self.QueryData()
        self.query_type = self.QueryData()
        self.answer = self.QueryData()
        self.ttl = self.QueryData()
```

B.7 /PalmaDNS/Framework/__init__.py

```
"""This is the init file for hte PalmaDNS package. It makes the directory
PalmaDNS/PalmaDNS accessable as a package. It will also import all
the main classes (core, plugins, queries, and query) and put those
modules in an __all__ variable as well.
"""

__all__ = ['core', 'plugins', 'queries', 'query']
from core import Core
from plugins import Plugins
from queries import Queries
from query import Query
```

B.8 /PalmaDNS/Framework/plugins.py

```
"""This file contains the Plugins() class and does not much more that give
this class the ability to function as a seperate module.
"""

try:
    from yapsy.PluginManager import PluginManager    # For analyzer plugins
    import logging                                   # For logging events
except ImportError, e:
    print 'Could not import module!'
    print 'ERROR - [' + str(e) + '] [' + __name__ + ']'
    raise SystemExit

class Plugins:
    """This is the class for having a simple interface to the plugin
    functionality of this framework. Simply put this class will act as
    a wrapper around the package Yet Another Plugin SYstem (yapsy).

    This class will define a plugin manager, set the given destiantion
    for where the plugins are stored. It will also deal with the activation
    of all or selective alarms, and to give other instances access to
    the names of the plugins and which ones are activated
    """

    def __init__(self, plugin_path):
        """Starts up the object by creating a logger for debugging and error
        messages. Also creates and initiates the PluginManager() that is
        yapsy object.
        @param plugin_path The full path for the directory where the plugins
```

```
29          and their respective configurations are located.
30          """
31          self.__logger = logging.getLogger(__name__)
32          self.__manager = PluginManager()
33          self.__manager.setPluginPlaces([plugin_path])
34          self.__manager.collectPlugins()
35
36      @property
37      def all_plugins(self):
38          """Return all the plugins that are configured in the PluginManager.
39          @retval plugins All the plugin objects that are in the Plugin Manager.
40          """
41          self.__logger.debug('Returned all plugins')
42          return self.__manager.getAllPlugins()
43
44      @property
45      def names(self):
46          """Return the names of all the plugins that are configured in the
47          PluginManager.
48          @retval plugins All the names of the plugins that are in the Plugin
49          Manager.
50          """
51          self.__logger.debug('Returned all plugin names')
52          names = [p.name for p in self.__manager.getAllPlugins()]
53          return names
54
55      @property
56      def activated(self):
57          """Return all the plugins that are  activated in the PluginManager.
58          @retval plugins All the activated plugins that are in the
59          Plugin Manager
60          """
61          self.__logger.debug('Returned all activated plugins')
62          plugins = [p for p in self.__manager.getAllPlugins() if p.is_activated]
63          return plugins
64
65      def activate(self, name):
66          """Activate a given plugin by its name, if a plugin with that name
67          cannot be found an logger.warning message is given.
68          @param name The name of the plugin a user wants to activate
69          """
70          if name in self.names:
71              self.__logger.debug('Activated plugin: %s' % name)
72              self.__manager.activatePluginByName(name)
73          else:
74              self.__logger.warning('Could not find plugin: %s' % name)
75
76      def activate_all(self):
77          """Activates all the plugins that are in the PluginMananger.
78          """
79          self.__logger.debug('Activated all plugins')
80          for name in self.names: self.__manager.activatePluginByName(name)
```

B.9 /PalmaDNS/Framework/config/logging.ini

```
1  [loggers]
2  keys = root
3
4  [handlers]
5  keys = consoleHandler
6
7  [formatters]
8  keys = simpleFormatter
9
10 [logger_root]
11 level = WARNING
12 handlers = consoleHandler
13
14 [handler_consoleHandler]
15 class = StreamHandler
16 level = DEBUG
17 formatter = simpleFormatter
18 args = (sys.stdout,)
19
20 [formatter_simpleFormatter]
21 format = %(asctime)s - %(name)s - %(levelname)s - %(message)s
22 datefmt =
```

B.10 /PalmaDNS/Framework/plugins/tld.py

```
"""tld.py contains the analysis plugin for TLD (Top Level Domain)."""

try:
  from yapsy.IPlugin import IPlugin
  import tldextract
except ImportError, e:
    print 'Could not import module!'
    print 'ERROR - [' + str(e) + '] [' + __name__ + ']'
    raise SystemExit

class TLD(IPlugin):
    """This is an analysis plugin for the PalmaDNS framework that analyses
    and determins if a TLD is suspicious or not.
    """

    def analyze(self, qobj):
        """Analysis function that does all the work of this analysis.
        @param qobj The query object for the query that will be analyzed.
        @retval result A float value between 0.0 and 10.0 indicating the
        evaluated maliciousnes of the query (10.0 == Bad)
        """
        tld_list = ['info', 'vn', 'cm', 'ms', 'am', 'com.co']
        ext = tldextract.extract(qobj.query.value)
        tld = ext.suffix
        if tld in tld_list:
            return 10.0
        return 1.0
```

B.11 /PalmaDNS/Framework/plugins/tld.yapsy-plugin

```
[Core]
Name = 001-TLD
Module = tld

[Documentation]
Author = Lars Christian Andersen & Andreas Moe
Version = 0.0.1
Website = http://hovedprosjekter.hig.no/v2014/imt/is/palmadns/
Description = An analyzer plugin to the PalmaDNS framework for detecting potential
     malicious TLD's in domain names
```

B.12 /PalmaDNS/Framework/plugins/ttl.py

```
"""ttl.py contains the analysis plugin for TTL (Time-To-Live)."""

try:
    from yapsy.IPlugin import IPlugin
except ImportError, e:
    print 'Could not import module!'
    print 'ERROR - [' + str(e) + '] [' + __name__ + ']'
    raise SystemExit

class TTL(IPlugin):
    """This is an analysis plugin for the PalmaDNS framework that analyses
    and determins if a TTL value is suspicious or not.
    """

    def analyze(self, qobj):
        """Analysis function that does all the work of this analysis.
        @param qobj The query object for the query that will be analyzed.
        @retval result A float value between 0.0 and 10.0 indicating the
        evaluated maliciousnes of the query (10.0 == Bad)
        """
        results = 1.0
        try:
            ttl = int(qobj.ttl.value)
            if ttl <= 100:
                results = 10.0
        except:
            pass

        return results
```

B.13 /PalmaDNS/Framework/plugins/ttl.yapsy-plugin

```
[Core]
Name = 002-TTL
Module = ttl

[Documentation]
Author = Andreas Moe & Lars Christian Andersen
Version = 0.0.1
Website = http://hovedprosjekter.hig.no/v2014/imt/is/palmadns/
Description = An analyzer plugin to the PalmaDNS framework for detecting potential
    malicious TTL values
```

B.14 /PalmaDNS/Framework/plugins/sample_plugin.py

```
"""Docstring for this file"""

try:
    from yapsy.IPlugin import IPlugin
except ImportError, e:
    print 'Could not import module!'
    print 'ERROR - [' + str(e) + '] [' + __name__ + ']'
    raise SystemExit

class SamplePlugin(IPlugin):
    """Docstring about this class"""

    def analyze(self, qobj):
        """Docstring about this function.
        - Extract wanted data from the QueryObject "qobj"
        - Do wizzardmagic
        - Return a score between 0.0 and 10.0 (10.0 == BAD)
        """
        pass
```

B.15 /PalmaDNS/README.txt

```
=== PalmaDNS ===
Authors: Andreas Moe & Lars Christian Andersen
License: See LICENSE.txt

PalmaDNS is a framework for plugin based analysis of passive dns data (pdns).

== Description ==
The PalmaDNS reads loggfiles of pdns sources of this kind:
  - https://github.com/gamelinux/passivedns

Plugins used for analysis are found in: PalmaDNS/PalmaDNS/plugins/. The plugins
follow the Yet Another Plugin SYstem (YAPSY) module. See sample_plugin.py for
examples of how to analyze data. The Framework can read from a file or
arguments given at execution in the commandline interface (CLI). Simply put the
framework wil read in data,  take each single DNS-query/response that is found
in the log/input, create an object for it, push all those objects onto a double
ended queue (deque). Then run an analysis where a query object is poped out of
the deque and analyzed in every plugin that has been defined.

For more information about the program please review the code and the
docstrings contained within, or if you understand Norwegian please read
our undergraduate thesis at:
  - http://hovedprosjekter.hig.no/v2014/imt/is/palmadns/index.html
```

B.16 /PalmaDNS/LICENSE.txt

```
The MIT License (MIT)

Copyright (c) 2014 Andreas Moe and Lars Christian Andersen

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

B.17 Dokumentasjon for PalmaDNS

PalmaDNS

0.1

Generated by Doxygen 1.8.6

Mon Apr 28 2014 11:35:12

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 PalmaDNS Namespace Reference

The arguments.py file is the module that holds the Arguments class which does all the work with initializing and gathering commandline arguments from the user.

Namespaces

- __main__
- arguments
- Framework

Variables

- list __all__ = ['Framework']

  *The PalmaDNS root folder is a container for the PalmaDNS package and for the CLI capabilities given through **main**.py and arguments.py.*

5.1.1 Detailed Description

The arguments.py file is the module that holds the Arguments class which does all the work with initializing and gathering commandline arguments from the user. There is no other function to this file than to include the needed modules, and to hold the Arguments class. See the arguments class docs for more information.

arguments.py

Author

Andreas Moe & Lars C. Andersen

5.1.2 Variable Documentation

5.1.2.1 list PalmaDNS.__all__ = ['Framework']

The PalmaDNS root folder is a container for the PalmaDNS package and for the CLI capabilities given through **main**.py and arguments.py.

5.2 PalmaDNS.__main__ Namespace Reference

Functions

- def main

  This is the main() function that is run when this file is executed from the shell.

Variables

- string __author__ = "Andreas Moe and Lars Christian Andersen"

  This is the main file in the package that is PlamaDNS, when this file is run the main() function is exectued.
- string __copyright__ = "Copyright 2014"
- string __license__ = "The MIT License (MIT)"
- string __version__ = "0.1"

5.2.1 Function Documentation

5.2.1.1 def PalmaDNS.__main__.main ( )

This is the main() function that is run when this file is executed from the shell.

This file will set ut the logging configuration, read and parse the commandline arguments through the Arguments() class, after that this function will give the user the wanted action they were after. Either showing the user more information about the program or running an analysis.

5.2.2 Variable Documentation

5.2.2.1 string PalmaDNS.__main__.__author__ = "Andreas Moe and Lars Christian Andersen"

This is the main file in the package that is PlamaDNS, when this file is run the main() function is exectued.

When this happens potential arguments are parsed and validated (see framework/arguments.py). A Framework object is also created (see framework/core.py). After that flow is directed to where and what the user wanted. This can be:

- Show information about the framework
- Analyze a file
- Analyze commandline arguments (domain, ip, etc given at execution)

5.2.2.2 string PalmaDNS.__main__.__copyright__ = "Copyright 2014"

5.2.2.3 string PalmaDNS.__main__.__license__ = "The MIT License (MIT)"

5.2.2.4 string PalmaDNS.__main__.__version__ = "0.1"

5.3 PalmaDNS.arguments Namespace Reference

Classes

- class Arguments

  This is the class that delivers all the arguments handeling and parsing to this framework.

5.4 PalmaDNS.Framework Namespace Reference

Namespaces

- core
- plugins
- queries
- query

Variables

- list __all__ = ['core', 'plugins', 'queries', 'query']

  This is the init file for hte PalmaDNS package.

5.4.1 Variable Documentation

5.4.1.1 list PalmaDNS.Framework.__all__ = ['core', 'plugins', 'queries', 'query']

This is the init file for hte PalmaDNS package.

It makes the directory PalmaDNS/PalmaDNS accessable as a package. It will also import all the main classes (core, plugins, queries, and query) and put those modules in an **all** variable as well.

5.5 PalmaDNS.Framework.core Namespace Reference

Classes

- class Core

  This is the framework class in core.py.

Variables

- string SOURCE_FILE = 'file'
- string SOURCE_PARAMS = 'params'
- string DEST_FILE = 'file'
- string DEST_SCREEN = 'screen'
- string DEST_RETURN = 'return'

5.5.1 Variable Documentation

5.5.1.1 string PalmaDNS.Framework.core.DEST_FILE = 'file'

5.5.1.2 string PalmaDNS.Framework.core.DEST_RETURN = 'return'

5.5.1.3 string PalmaDNS.Framework.core.DEST_SCREEN = 'screen'

5.5.1.4 string PalmaDNS.Framework.core.SOURCE_FILE = 'file'

5.5.1.5 string PalmaDNS.Framework.core.SOURCE_PARAMS = 'params'

5.6 PalmaDNS.Framework.plugins Namespace Reference

Classes

- class Plugins

  This is the class for having a simple interface to the plugin functionality of this framework.

5.7 PalmaDNS.Framework.queries Namespace Reference

Classes

- class Queries

  This is the class for holding all the Passive DNS detected queries that the framework is to anlyze.

5.8 PalmaDNS.Framework.query Namespace Reference

Classes

- class Query

  Holds the Query() class, see the Query docstring for more information.

5.9 sample_plugin Namespace Reference

Classes

- class SamplePlugin

  Docstring about this class.

5.10 tld Namespace Reference

Classes

- class TLD

  This is an analysis plugin for the PalmaDNS framework that analyses and determins if a TLD is suspicious or not.

5.11 ttl Namespace Reference

Classes

- class TTL

  This is an analysis plugin for the PalmaDNS framework that analyses and determins if a TTL value is suspicious or not.

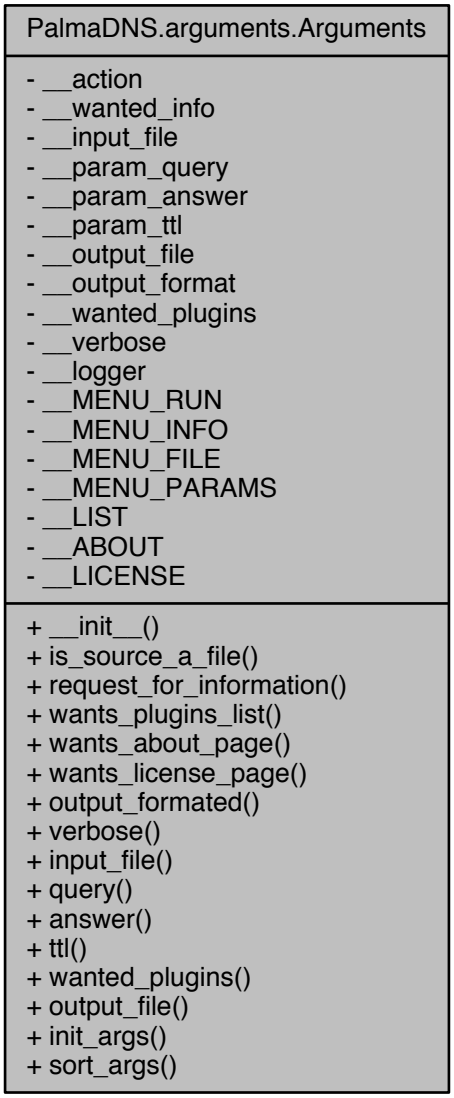
Chapter 6

Class Documentation

6.1 PalmaDNS.arguments.Arguments Class Reference

This is the class that delivers all the arguments handeling and parsing to this framework.

Collaboration diagram for PalmaDNS.arguments.Arguments:



Public Member Functions

- def __init__

  Initiates all private variables and runs initialization and sorting of commandline arguments.
- def is_source_a_file

  Checks if the datasource is a file.
- def request_for_information

  Checks to see if the user wanted to see more information about this program and returns the answer as a boolean True/False.
- def wants_plugins_list

  Checks if the user wanted to see a list of all the plugins that have been added to the 'plugins' directory and returns the answer as a boolean True/False.

- def wants_about_page

  Checks to see if the user wanted to be shown the 'about' page for this program (which is the README.txt file) and returns the answer as a boolean True/False.

- def wants_license_page

  Checks to see if the user wanted to show the license information for this program (which is the LICENSE.txt file) and returns the answer as a boolean True/False.

- def output_formated

  Returns the value of wether or not the ouput from the future analysis should be given as a raw data format or a stylized format that is more human readable.

- def verbose

  Returns if the user wants verbose output while running the program.

- def input_file

  Returns what input file the user has given to the program.

- def query

  Returns the 'query' value that the user gave to the program as a commandline argument if the user is about to analyze data that is not from a file.

- def answer

  Returns the 'answer' value that the user gave to the program as a commandline argument if the user is about to analyze data that is not from a file.

- def ttl

  Returns the 'Time-To-Live' value that the user gave to the program as a commandline argument if the user is about to analyze data that is not from a file.

- def wanted_plugins

  Returnes the specified plugins that the user wants to run with this program.

- def output_file

  Returns the output file that the user (potentialy) has given to indicate where to program should write its results.

- def init_args

  This method sets up the arguments parsers for the different parts of functionality the the framework can deliver.

- def sort_args

  This method is for sorting through the given arguemnts and put them into the common datastructure so that they are easly accessable for the data methods in this class so that the main framework can see what the user wanted to be done.

Private Attributes

- __action
- __wanted_info
- __input_file
- __param_query
- __param_answer
- __param_ttl
- __output_file
- __output_format
- __wanted_plugins
- __verbose
- __logger

Static Private Attributes

- string __MENU_RUN = 'run'
- string __MENU_INFO = 'information'
- string __MENU_FILE = 'file'
- string __MENU_PARAMS = 'params'
- string __LIST = 'list'
- string __ABOUT = 'about'
- string __LICENSE = 'license'

6.1.1 Detailed Description

This is the class that delivers all the arguments handeling and parsing to this framework.

This class does mainly three things: 1: Initiates an arguments parser and defines the different arguments that can be set, given or that can / cannot co-exist. This is done in the self.init_args() method

2: Sortes through the arguments and sets the recived data into a data- structure (self.__data) so that is is easily accessable with differenct methods for the framework to decide what the user wanted This is done in the self.sort_-args() method

3: Storing and giving out data about the arguments the user gave as commandline arguments. The results of part 2 above are stored in a common datastructure (self.__data) and this class contains a serifes of checking methods (eg. check if fancy formatting) is requested. And some property methods (eg. instance.input_file), so that theese datas are avaiable for the framework

6.1.2 Constructor & Destructor Documentation

6.1.2.1 def PalmaDNS.arguments.Arguments.__init__ (*self*)

Initiates all private variables and runs initialization and sorting of commandline arguments.

Sets all the private variables with regard to the functionality the user wanted to 'None'. A logger is created to handle error and debugging messages. The args are initialized and sorted at the end of the constructor.

6.1.3 Member Function Documentation

6.1.3.1 def PalmaDNS.arguments.Arguments.answer (*self*)

Returns the 'answer' value that the user gave to the program as a commandline argument if the user is about to analyze data that is not from a file.

6.1.3.2 def PalmaDNS.arguments.Arguments.init_args (*self*)

This method sets up the arguments parsers for the different parts of functionality the the framework can deliver.

The different parsers / functionality is:

Parser: The main parser. parser_analyze: For the arguments that can be given with regards to analyzing data.

parser_info: For the arguments for displaying more information about the framework.

parser_analyze_file: The parser for arguments regarding file analysis.

parser_analyze_input: The parser for arguments regarding analysis of commandline input / arguments

```
Getting more information:   ./palma.py information ...
Analyzing something:        ./palma.py run ...
Analyzing a file:           ./palma.py run file ...
Analysing paramters:        ./palma.py run params ...
```

6.1.3.3 def PalmaDNS.arguments.Arguments.input_file (*self*)

Returns what input file the user has given to the program.

If the program is going to analyze input parameters this will by default be set to null due to the constructor.

6.1.3.4 def PalmaDNS.arguments.Arguments.is_source_a_file (*self*)

Checks if the datasource is a file.

Checks to see if the source of the data that is to be analyzed is given from a file and returns the answer as a boolean True/False.

6.1.3.5 def PalmaDNS.arguments.Arguments.output_file (*self*)

Returns the output file that the user (potentialy) has given to indicate where to program should write its results.

6.1.3.6 def PalmaDNS.arguments.Arguments.output_formated (*self*)

Returns the value of wether or not the ouput from the future analysis should be given as a raw data format or a stylized format that is more human readable.

6.1.3.7 def PalmaDNS.arguments.Arguments.query (*self*)

Returns the 'query' value that the user gave to the program as a commandline argument if the user is about to analyze data that is not from a file.

6.1.3.8 def PalmaDNS.arguments.Arguments.request_for_information (*self*)

Checks to see if the user wanted to see more information about this program and returns the answer as a boolean True/False.

6.1.3.9 def PalmaDNS.arguments.Arguments.sort_args (*self, args*)

This method is for sorting through the given arguemnts and put them into the common datastructure so that they are easly accessable for the data methods in this class so that the main framework can see what the user wanted to be done.

Parameters

args	The arguments object containing the commandline arguments given by the user at execution time.

6.1.3.10 def PalmaDNS.arguments.Arguments.ttl (*self*)

Returns the 'Time-To-Live' value that the user gave to the program as a commandline argument if the user is about to analyze data that is not from a file.

6.1.3.11 def PalmaDNS.arguments.Arguments.verbose (*self*)

Returns if the user wants verbose output while running the program.

This is used for when users want more information about the running of the program and debugging purposes. This will activate all logging events at the level 'DEBUG'.

6.1.3.12 def PalmaDNS.arguments.Arguments.wanted_plugins (*self*)

Returnes the specified plugins that the user wants to run with this program.

6.1.3.13 def PalmaDNS.arguments.Arguments.wants_about_page (*self*)

Checks to see if the user wanted to be shown the 'about' page for this program (which is the README.txt file) and returns the answer as a boolean True/False.

6.1.3.14 def PalmaDNS.arguments.Arguments.wants_license_page (*self*)

Checks to see if the user wanted to show the license information for this program (which is the LICENSE.txt file) and returns the answer as a boolean True/False.

6.1.3.15 def PalmaDNS.arguments.Arguments.wants_plugins_list (*self*)

Checks if the user wanted to see a list of all the plugins that have been added to the 'plugins' directory and returns the answer as a boolean True/False.

6.1.4 Member Data Documentation

6.1.4.1 string PalmaDNS.arguments.Arguments.__ABOUT = 'about' [static], [private]

6.1.4.2 PalmaDNS.arguments.Arguments.__action [private]

6.1.4.3 PalmaDNS.arguments.Arguments.__input_file [private]

6.1.4.4 string PalmaDNS.arguments.Arguments.__LICENSE = 'license' [static], [private]

6.1.4.5 string PalmaDNS.arguments.Arguments.__LIST = 'list' [static], [private]

6.1.4.6 PalmaDNS.arguments.Arguments.__logger [private]

6.1.4.7 string PalmaDNS.arguments.Arguments.__MENU_FILE = 'file' [static], [private]

6.1.4.8 string PalmaDNS.arguments.Arguments.__MENU_INFO = 'information' [static], [private]

6.1.4.9 string PalmaDNS.arguments.Arguments.__MENU_PARAMS = 'params' [static], [private]

6.1.4.10 string PalmaDNS.arguments.Arguments.__MENU_RUN = 'run' [static], [private]

6.1.4.11 PalmaDNS.arguments.Arguments.__output_file [private]

6.1.4.12 PalmaDNS.arguments.Arguments.__output_format [private]

6.1.4.13 PalmaDNS.arguments.Arguments.__param_answer [private]

6.1.4.14 PalmaDNS.arguments.Arguments.__param_query [private]

6.1.4.15 PalmaDNS.arguments.Arguments.__param_ttl [private]

6.1.4.16 PalmaDNS.arguments.Arguments.__verbose [private]

6.1.4.17 PalmaDNS.arguments.Arguments.__wanted_info [private]

6.1.4.18 PalmaDNS.arguments.Arguments.__wanted_plugins [private]

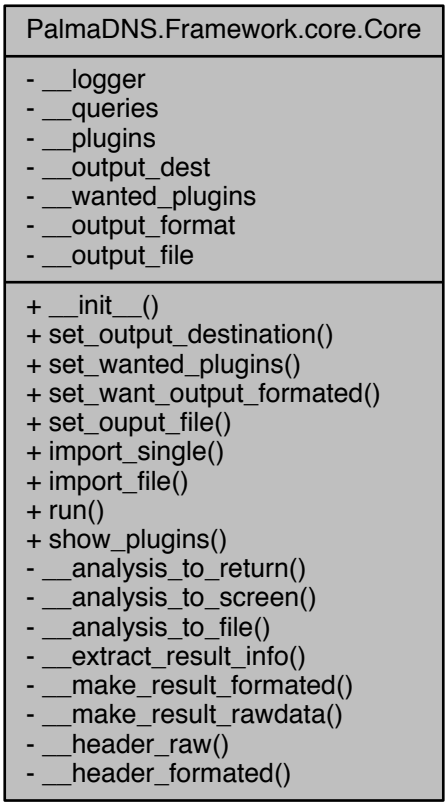
The documentation for this class was generated from the following file:

- Programmering/palmadns/Code/PalmaDNS/arguments.py

6.2 PalmaDNS.Framework.core.Core Class Reference

This is the framework class in core.py.

Collaboration diagram for PalmaDNS.Framework.core.Core:



Public Member Functions

- def __init__

  Initiates the object by firstly creating a logger to handle potential debugging and error messages.
- def set_output_destination

  Sets the wanted output destination (file, screen, return).
- def set_wanted_plugins

  Sets the wanted plugins that should be used in an analysis.

- def set_want_output_formated

  Sets the wanted output format.
- def set_ouput_file

  Sets the output file is the destination is going to be a file.
- def import_single

  This method is used to import a single pdns query.
- def import_file

  This method is used to import a pdns log file.
- def run

  This is the function that initializes the framework to run an analysis.
- def show_plugins

  Shows a formated printout (screen) of all the available plugins.

Private Member Functions

- def __analysis_to_return

  Runs through the queries deque and runs all plugins on given query, then formats accoring to format_func choice and writes the results to screen.
- def __analysis_to_screen

  Runs through the queries deque and runs all plugins on given query, then formats accoring to format_func choice and writes the results to screen.
- def __analysis_to_file

  Runs through the queries deque and runs all plugins on given query, then formats accoring to format_func choice and writes the results to file.
- def __extract_result_info

  Exctracts the wanted information needed for generating an alarm.
- def __make_result_formated

  Creates a formated and 'fancy' printout of each alarm result.
- def __make_result_rawdata

  Extracts information about this alarm and joins to a CSV format.
- def __header_raw

  Creates a header string for when outout is rawdata results.
- def __header_formated

  Creates a header string for when output is formated.

Private Attributes

- __logger
- __queries
- __plugins
- __output_dest
- __wanted_plugins
- __output_format
- __output_file

6.2.1 Detailed Description

This is the framework class in core.py.

The role of this class is to be the glue between all the components of this framework. This class will hold and administrate the Queries, Plugins and Arguments parseing. Do the file / screen writing of results and messages to the user. This class will also do the analysis by emptying the Queries deque, and then runing that query object into each activaed plugin, and divert the results to the wanted destination (file or screen) and in the wanted format.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 def PalmaDNS.Framework.core.Core.__init__ (*self*)

Initiates the object by firstly creating a logger to handle potential debugging and error messages.

Afther this a Queries() object is created to handle all the individual queries (that will be imported at a later time). Then a Plugins() object is created and a path the the plugins folder is suplied. After that some common variables about the input / ouput configuration are created and set to None.

6.2.3 Member Function Documentation

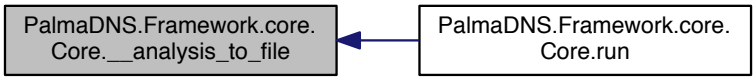
6.2.3.1 def PalmaDNS.Framework.core.Core.__analysis_to_file (*self, plugins, format_fun, header_fun, ofile*) [private]

Runs through the queries deque and runs all plugins on given query, then formats accoring to format_func choice and writes the results to file.

Return values

results	Always 'None' since output goes to a file.

Here is the caller graph for this function:



6.2.3.2 def PalmaDNS.Framework.core.Core.__analysis_to_return (*self, plugins, format_fun, header_fun, dummy =* `None`) [private]

Runs through the queries deque and runs all plugins on given query, then formats accoring to format_func choice and writes the results to screen.

Here is the caller graph for this function:

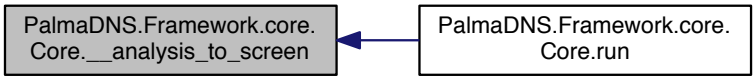
6.2.3.3 def PalmaDNS.Framework.core.Core.__analysis_to_screen (*self, plugins, format_fun, header_fun, dummy =* `None`) [private]

Runs through the queries deque and runs all plugins on given query, then formats accoring to format_func choice and writes the results to screen.

Return values

results	Always 'None' since output goes to the screen.

Here is the caller graph for this function:



6.2.3.4 def PalmaDNS.Framework.core.Core.__extract_result_info (*self, result, alarm, qobj*) [private]

Exctracts the wanted information needed for generating an alarm.

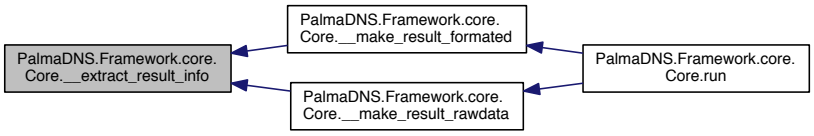
Parameters

result	The result of a plugins analysis function.
alarm	The name of the plugin that has been run.
qobj	The query object that was analzed (See class Query()).

Return values

result	A list containing all the data.

Here is the caller graph for this function:



6.2.3.5 def PalmaDNS.Framework.core.Core.__header_formated (*self*) [private]

Creates a header string for when output is formated.

Return values

header	A formated header string.

Here is the caller graph for this function:

6.2.3.6 def PalmaDNS.Framework.core.Core.__header_raw (*self*) [private]

Creates a header string for when outout is rawdata results.

Return values

header	A CSV foramted header string.

Here is the caller graph for this function:

6.2.3.7 def PalmaDNS.Framework.core.Core.__make_result_formated (*self, result, alarm, qobj*) [private]

Creates a formated and 'fancy' printout of each alarm result.

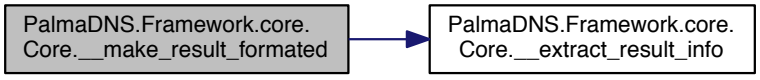
Parameters

result	The result of a plugins analysis function.
alarm	The name of the plugin that has been run.
qobj	The query object that was analzed (See class Query()).

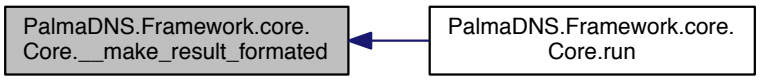
Return values

result	A formated string with the reults of this analysis.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.3.8 def PalmaDNS.Framework.core.Core.__make_result_rawdata (*self, result, alarm, qobj*) [private]

Extracts information about this alarm and joins to a CSV format.

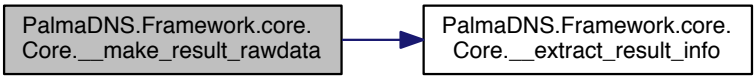
Parameters

result	The result of a plugins analysis function.
alarm	The name of the plugin that has been run.
qobj	The query object that was analzed (See class Query()).

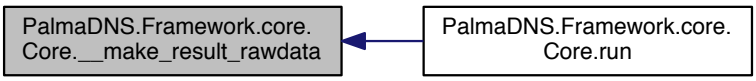
Return values

result	A CSV formated string.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.3.9 def PalmaDNS.Framework.core.Core.import_file (*self, input_file*)

This method is used to import a pdns log file.

Parameters

input_file	The file that is going to be imported.

6.2.3.10 def PalmaDNS.Framework.core.Core.import_single (*self, query, answer, ttl*)

This method is used to import a single pdns query.

Parameters

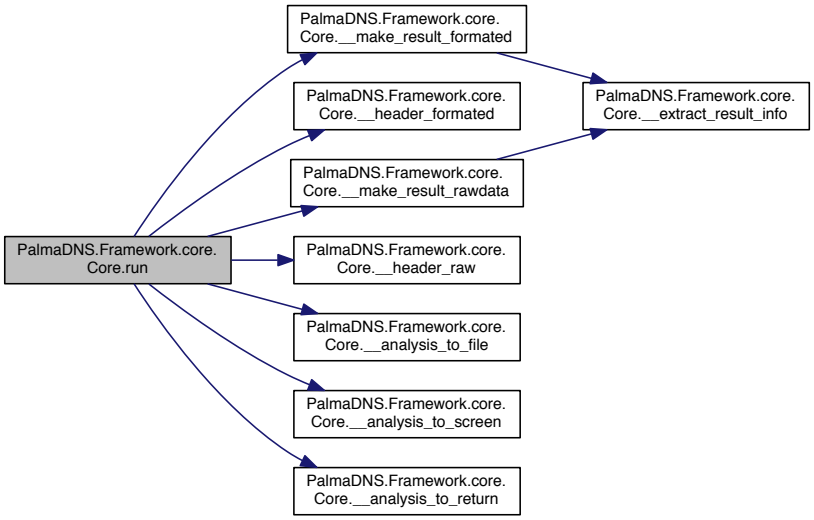
query	The domain name that was queried
answer	What the query resolved to
ttl	The Time-To-Live for this resolve

6.2.3.11 def PalmaDNS.Framework.core.Core.run (*self*)

This is the function that initializes the framework to run an analysis.

The plugins are activated (if any are specified). The type of analysis output if figured out, and the formatting function is also chosen. After all that the desired analysis with its given output destination is run. Happy analysis!

Here is the call graph for this function:



6.2.3.12 def PalmaDNS.Framework.core.Core.set_ouput_file (*self, file*)

Sets the output file is the destination is going to be a file.

Parameters

file	Sets the destination file for the analysis results

6.2.3.13 def PalmaDNS.Framework.core.Core.set_output_destination (*self, destination*)

Sets the wanted output destination (file, screen, return).

Parameters

destination	The destination for the output of this analysis.

6.2.3.14 def PalmaDNS.Framework.core.Core.set_want_output_formated (*self, output_format*)

Sets the wanted output format.

Parameters

output_format	Sets the type of output format the user wants.

6.2.3.15 def PalmaDNS.Framework.core.Core.set_wanted_plugins (*self, plugins*)

Sets the wanted plugins that should be used in an analysis.

Parameters

plugins	The plugins the user wants to use for this analysis.

6.2.3.16 def PalmaDNS.Framework.core.Core.show_plugins (*self*)

Shows a formated printout (screen) of all the available plugins.

6.2.4 Member Data Documentation

6.2.4.1 PalmaDNS.Framework.core.Core.__logger [private]

6.2.4.2 PalmaDNS.Framework.core.Core.__output_dest [private]

6.2.4.3 PalmaDNS.Framework.core.Core.__output_file [private]

6.2.4.4 PalmaDNS.Framework.core.Core.__output_format [private]

6.2.4.5 PalmaDNS.Framework.core.Core.__plugins [private]

6.2.4.6 PalmaDNS.Framework.core.Core.__queries [private]

6.2.4.7 PalmaDNS.Framework.core.Core.__wanted_plugins [private]

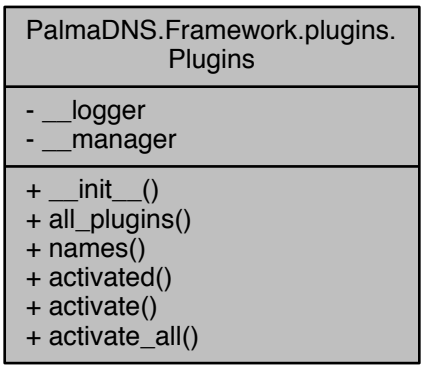
The documentation for this class was generated from the following file:

- Programmering/palmadns/Code/PalmaDNS/Framework/core.py

6.3 PalmaDNS.Framework.plugins.Plugins Class Reference

This is the class for having a simple interface to the plugin functionality of this framework.

Collaboration diagram for PalmaDNS.Framework.plugins.Plugins:



Public Member Functions

- def __init__

  Starts up the object by creating a logger for debugging and error messages.
- def all_plugins

  Return all the plugins that are configured in the PluginManager.
- def names

  Return the names of all the plugins that are configured in the PluginManager.
- def activated

  Return all the plugins that are activated in the PluginManager.
- def activate

  Activate a given plugin by its name, if a plugin with that name cannot be found an logger.warning message is given.
- def activate_all

  Activates all the plugins that are in the PluginMananger.

Private Attributes

- __logger
- __manager

6.3.1 Detailed Description

This is the class for having a simple interface to the plugin functionality of this framework.

Simply put this class will act as a wrapper around the package Yet Another Plugin SYstem (yapsy).

This class will define a plugin manager, set the given destiantion for where the plugins are stored. It will also deal with the activation of all or selective alarms, and to give other instances access to the names of the plugins and which ones are activated

6.3.2 Constructor & Destructor Documentation

6.3.2.1 def PalmaDNS.Framework.plugins.Plugins.__init__ (*self, plugin_path*)

Starts up the object by creating a logger for debugging and error messages.

Also creates and initiates the PluginManager() that is yapsy object.

Parameters

plugin_path	The full path for the directory where the plugins and their respective configurations are located.

6.3.3 Member Function Documentation

6.3.3.1 def PalmaDNS.Framework.plugins.Plugins.activate (*self, name*)

Activate a given plugin by its name, if a plugin with that name cannot be found an logger.warning message is given.

Parameters

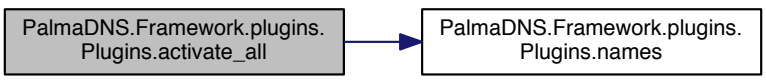
name	The name of the plugin a user wants to activate

Here is the call graph for this function:

6.3.3.2 def PalmaDNS.Framework.plugins.Plugins.activate_all (*self*)

Activates all the plugins that are in the PluginMananger.

Here is the call graph for this function:



6.3.3.3 def PalmaDNS.Framework.plugins.Plugins.activated (*self*)

Return all the plugins that are activated in the PluginManager.

Return values

plugins	All the activated plugins that are in the Plugin Manager

6.3.3.4 def PalmaDNS.Framework.plugins.Plugins.all_plugins (*self*)

Return all the plugins that are configured in the PluginManager.

Return values

plugins	All the plugin objects that are in the Plugin Manager.

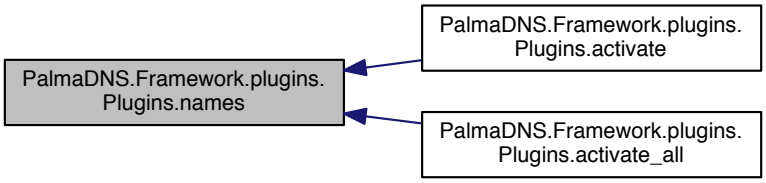
6.3.3.5 def PalmaDNS.Framework.plugins.Plugins.names (*self*)

Return the names of all the plugins that are configured in the PluginManager.

Return values

plugins	All the names of the plugins that are in the Plugin Manager.

Here is the caller graph for this function:



6.3.4 Member Data Documentation

6.3.4.1 PalmaDNS.Framework.plugins.Plugins.__logger [private]

6.3.4.2 PalmaDNS.Framework.plugins.Plugins.__manager [private]

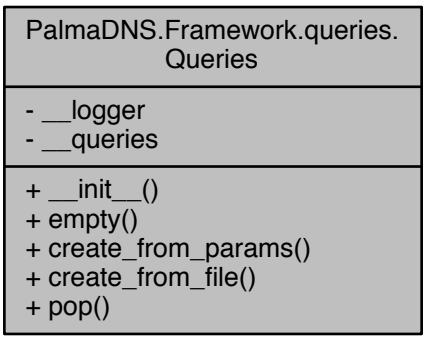
The documentation for this class was generated from the following file:

- Programmering/palmadns/Code/PalmaDNS/Framework/plugins.py

6.4 PalmaDNS.Framework.queries.Queries Class Reference

This is the class for holding all the Passive DNS detected queries that the framework is to anlyze.

Collaboration diagram for PalmaDNS.Framework.queries.Queries:



Public Member Functions

- def __init__

  Initializes the object by creating a logger and a double ended queue for holding individual query objects.
- def empty

  This method is just a simple check to se if there is any data in the deque __data.
- def create_from_params

  This method puts the given parameters (domain, ip, ttl) and puts them into a query object that is appended to the deque __data.
- def create_from_file

  This method reads the contents of a CSV file that contains passive dns captured data.
- def pop

  This method gets the first element (to the left) with a .popleft() and returns it to the calling function / user.

Private Attributes

- __logger
- __queries

6.4.1 Detailed Description

This is the class for holding all the Passive DNS detected queries that the framework is to anlyze.

The concept here is to have a common interface to creating, storing and retriving information about the queries.

This class will read single queries or read them from file. The data is then stored in a Query object (see /framework/query.py) and that obejct is added to a double ended queue (collections.deque). This is done so that the first element to be added can be analyzed first and that this process should go fast (lists would be slower, etc).

6.4.2 Constructor & Destructor Documentation

6.4.2.1 def PalmaDNS.Framework.queries.Queries.__init__ (*self*)

Initializes the object by creating a logger and a double ended queue for holding individual query objects.

6.4.3 Member Function Documentation

6.4.3.1 def PalmaDNS.Framework.queries.Queries.create_from_file (*self, filename*)

This method reads the contents of a CSV file that contains passive dns captured data.

Each line is read with the help of the csv.reader method. Each line is then put into a query object and the appended to the deque __data.

The format for this CSV file must be (header): timestamp,dns_client,dns_server,rr_class,query,query_-type,answer,ttl

timestamp = The timestamp of the event (epoch) dns_client = The client that performed this query dns_server = The server that answer this query rr_class = The class of the Resource Record query = The data (normaly domain) that was queried about query_typ = The type of query response that was given (A, PRT, etc) answer = What did the query resolve to ttl = The time to live of this query response

Parameters

filename	The name for the file that this function reads its passive dns data from.

6.4.3.2 def PalmaDNS.Framework.queries.Queries.create_from_params (*self, query, answer, ttl*)

This method puts the given parameters (domain, ip, ttl) and puts them into a query object that is appended to the deque __data.

Parameters

query	The domain name that was queried
answer	What the query resolved to
ttl	The Time-To-Live for this resolve

6.4.3.3 def PalmaDNS.Framework.queries.Queries.empty (*self*)

This method is just a simple check to se if there is any data in the deque __data.

This is used when looping through all the elements or just simply to check the current status. This is configured as property so it can be called by: object.is_empty (as a variable)

Return values

__queries	Wether or not there are any queries in this object.

6.4.3.4 def PalmaDNS.Framework.queries.Queries.pop (*self*)

This method gets the first element (to the left) with a .popleft() and returns it to the calling function / user.

Return values

query	The first query in the double ended queue.

6.4.4 Member Data Documentation

6.4.4.1 PalmaDNS.Framework.queries.Queries.__logger [private]

6.4.4.2 PalmaDNS.Framework.queries.Queries.__queries [private]

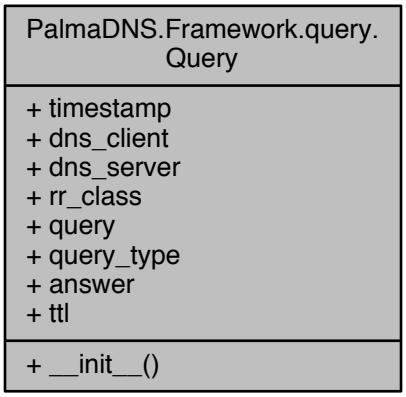
The documentation for this class was generated from the following file:

- Programmering/palmadns/Code/PalmaDNS/Framework/queries.py

6.5 PalmaDNS.Framework.query.Query Class Reference

Holds the Query() class, see the Query docstring for more information.

Collaboration diagram for PalmaDNS.Framework.query.Query:



Classes

- class QueryData

  This is a nested class to reduce the amount of code that had to be written.

Public Member Functions

- def __init__

  Initiates this object by creating a QueryData() variable for each of the different attributes that this Query will contain.

Public Attributes

- timestamp
- dns_client
- dns_server
- rr_class
- query
- query_type
- answer
- ttl

6.5.1 Detailed Description

Holds the Query() class, see the Query docstring for more information.

This is the class that gives the framework objects to be able to store in the queries class. Each instance of this object contains all the possible data that a passive DNS query capture can contain (according the the specifications of this framework). Theese elements are

timestamp = The timestamp of the event (epoch) dns_client = The client that performed this query dns_server = The server that answer this query rr_class = The class of the Resource Record query = The data (normaly domain) that was queried about query_typ = The type of query response that was given (A, PRT, etc) answer = What did the query resolve to ttl = The time to live of this query response

6.5.2 Constructor & Destructor Documentation

6.5.2.1 def PalmaDNS.Framework.query.Query.__init__ (*self*)

Initiates this object by creating a QueryData() variable for each of the different attributes that this Query will contain.

6.5.3 Member Data Documentation

6.5.3.1 PalmaDNS.Framework.query.Query.answer

6.5.3.2 PalmaDNS.Framework.query.Query.dns_client

6.5.3.3 PalmaDNS.Framework.query.Query.dns_server

6.5.3.4 PalmaDNS.Framework.query.Query.query

6.5.3.5 PalmaDNS.Framework.query.Query.query_type

6.5.3.6 PalmaDNS.Framework.query.Query.rr_class

6.5.3.7 PalmaDNS.Framework.query.Query.timestamp

6.5.3.8 PalmaDNS.Framework.query.Query.ttl

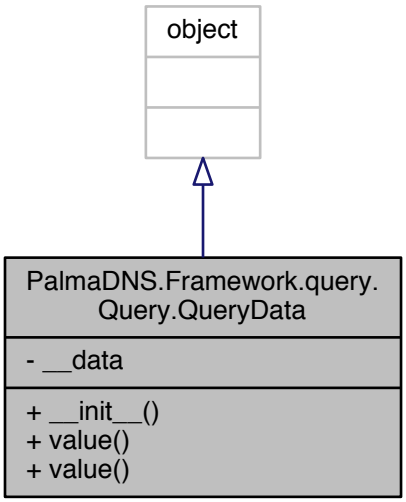
The documentation for this class was generated from the following file:

- Programmering/palmadns/Code/PalmaDNS/Framework/query.py

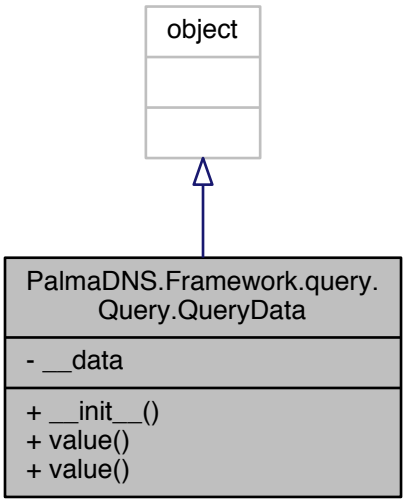
6.6 PalmaDNS.Framework.query.Query.QueryData Class Reference

This is a nested class to reduce the amount of code that had to be written.

Inheritance diagram for PalmaDNS.Framework.query.Query.QueryData:



Collaboration diagram for PalmaDNS.Framework.query.Query.QueryData:



Public Member Functions

- def __init__

  Create the data holder and set it default to None.
- def value

Returns the contents of the data holder to the caller.

- def value

  Sets the data holder to holde a new value.

Private Attributes

- __data

6.6.1 Detailed Description

This is a nested class to reduce the amount of code that had to be written.

The functionality is the same for all the variables so this class was created.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 def PalmaDNS.Framework.query.Query.QueryData.__init__ (*self*)

Create the data holder and set it default to None.

6.6.3 Member Function Documentation

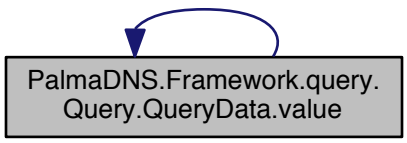
6.6.3.1 def PalmaDNS.Framework.query.Query.QueryData.value (*self*)

Returns the contents of the data holder to the caller.

Return values

__data	The data contained in this object

Here is the caller graph for this function:



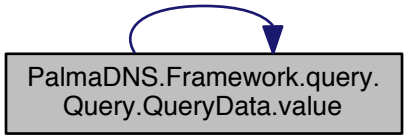
6.6.3.2 def PalmaDNS.Framework.query.Query.QueryData.value (*self, data*)

Sets the data holder to holde a new value.

Parameters

data	The new value for this data holder variable

Here is the call graph for this function:



6.6.4 Member Data Documentation

6.6.4.1 PalmaDNS.Framework.query.Query.QueryData.__data [private]

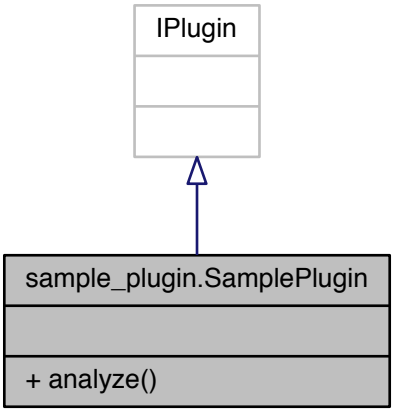
The documentation for this class was generated from the following file:

- Programmering/palmadns/Code/PalmaDNS/Framework/query.py

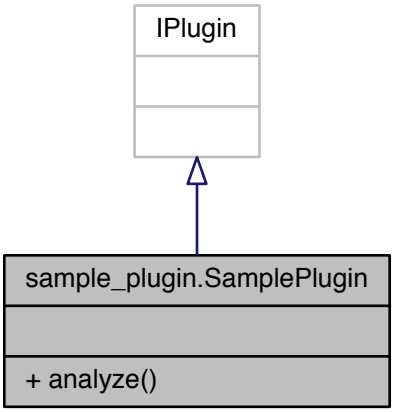
6.7 sample_plugin.SamplePlugin Class Reference

Docstring about this class.

Inheritance diagram for sample_plugin.SamplePlugin:



Collaboration diagram for sample_plugin.SamplePlugin:



Public Member Functions

- def analyze

  Docstring about this function.

6.7.1 Detailed Description

Docstring about this class.

6.7.2 Member Function Documentation

6.7.2.1 def sample_plugin.SamplePlugin.analyze (*self, qobj*)

Docstring about this function.

- Extract wanted data from the QueryObject "qobj"
- Do wizzardmagic
- Return a score between 0.0 and 10.0 (10.0 == BAD)

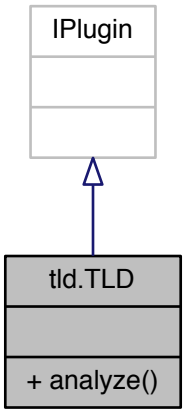
The documentation for this class was generated from the following file:

- Programmering/palmadns/Code/PalmaDNS/Framework/plugins/sample_plugin.py

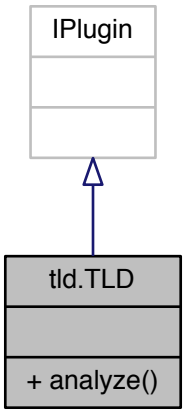
6.8 tld.TLD Class Reference

This is an analysis plugin for the PalmaDNS framework that analyses and determins if a TLD is suspicious or not.

Inheritance diagram for tld.TLD:



Collaboration diagram for tld.TLD:



Public Member Functions

- def analyze

  Analysis function that does all the work of this analysis.

6.8.1 Detailed Description

This is an analysis plugin for the PalmaDNS framework that analyses and determins if a TLD is suspicious or not.

6.8.2 Member Function Documentation

6.8.2.1 def tld.TLD.analyze (*self, qobj*)

Analysis function that does all the work of this analysis.

Parameters

qobj	The query object for the query that will be analyzed.

Return values

result	A float value between 0.0 and 10.0 indicating the evaluated maliciousnes of the query (10.0 == Bad)

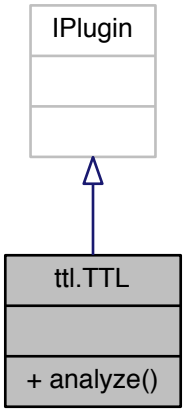
The documentation for this class was generated from the following file:

- Programmering/palmadns/Code/PalmaDNS/Framework/plugins/tld.py

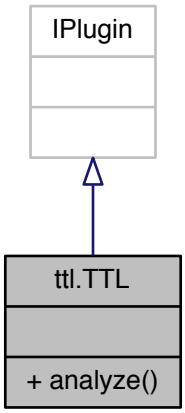
6.9 ttl.TTL Class Reference

This is an analysis plugin for the PalmaDNS framework that analyses and determins if a TTL value is suspicious or not.

Inheritance diagram for ttl.TTL:



Collaboration diagram for ttl.TTL:



Public Member Functions

- def analyze

  Analysis function that does all the work of this analysis.

6.9.1 Detailed Description

This is an analysis plugin for the PalmaDNS framework that analyses and determins if a TTL value is suspicious or not.

6.9.2 Member Function Documentation

6.9.2.1 def ttl.TTL.analyze (*self, qobj*)

Analysis function that does all the work of this analysis.

Parameters

qobj	The query object for the query that will be analyzed.

Return values

result	A float value between 0.0 and 10.0 indicating the evaluated maliciousnes of the query (10.0 == Bad)

The documentation for this class was generated from the following file:

- Programmering/palmadns/Code/PalmaDNS/Framework/plugins/ttl.py

Chapter 7

File Documentation

7.1 Programmering/palmadns/Code/PalmaDNS/__init__.py File Reference

Namespaces

- PalmaDNS

  The arguments.py file is the module that holds the Arguments class which does all the work with initializing and gathering commandline arguments from the user.

Variables

- list PalmaDNS.__all__ = ['Framework']

  *The PalmaDNS root folder is a container for the PalmaDNS package and for the CLI capabilities given through **main**.py and arguments.py.*

7.2 Programmering/palmadns/Code/PalmaDNS/Framework/__init__.py File Reference

Namespaces

- PalmaDNS.Framework

Variables

- list PalmaDNS.Framework.__all__ = ['core', 'plugins', 'queries', 'query']

  This is the init file for hte PalmaDNS package.

7.3 Programmering/palmadns/Code/PalmaDNS/__main__.py File Reference

Namespaces

- PalmaDNS.__main__

Functions

- def PalmaDNS.__main__.main

  This is the main() function that is run when this file is executed from the shell.

Variables

- string PalmaDNS.__main__.__author__ = "Andreas Moe and Lars Christian Andersen"

  This is the main file in the package that is PlamaDNS, when this file is run the main() function is exectued.
- string PalmaDNS.__main__.__copyright__ = "Copyright 2014"
- string PalmaDNS.__main__.__license__ = "The MIT License (MIT)"
- string PalmaDNS.__main__.__version__ = "0.1"

7.4 Programmering/palmadns/Code/PalmaDNS/arguments.py File Reference

Classes

- class PalmaDNS.arguments.Arguments

  This is the class that delivers all the arguments handeling and parsing to this framework.

Namespaces

- PalmaDNS.arguments
- PalmaDNS

  The arguments.py file is the module that holds the Arguments class which does all the work with initializing and gathering commandline arguments from the user.

7.5 Programmering/palmadns/Code/PalmaDNS/Framework/core.py File Reference

Classes

- class PalmaDNS.Framework.core.Core

  This is the framework class in core.py.

Namespaces

- PalmaDNS.Framework.core

Variables

- string PalmaDNS.Framework.core.SOURCE_FILE = 'file'
- string PalmaDNS.Framework.core.SOURCE_PARAMS = 'params'
- string PalmaDNS.Framework.core.DEST_FILE = 'file'
- string PalmaDNS.Framework.core.DEST_SCREEN = 'screen'
- string PalmaDNS.Framework.core.DEST_RETURN = 'return'

7.6 Programmering/palmadns/Code/PalmaDNS/Framework/plugins.py File Reference

Classes

- class PalmaDNS.Framework.plugins.Plugins

  This is the class for having a simple interface to the plugin functionality of this framework.

Namespaces

- PalmaDNS.Framework.plugins

7.7 Programmering/palmadns/Code/PalmaDNS/Framework/plugins/sample_plugin.py File Reference

Classes

- class sample_plugin.SamplePlugin

  Docstring about this class.

Namespaces

- sample_plugin

7.8 Programmering/palmadns/Code/PalmaDNS/Framework/plugins/tld.py File Reference

Classes

- class tld.TLD

  This is an analysis plugin for the PalmaDNS framework that analyses and determins if a TLD is suspicious or not.

Namespaces

- tld

7.9 Programmering/palmadns/Code/PalmaDNS/Framework/plugins/ttl.py File Reference

Classes

- class ttl.TTL

  This is an analysis plugin for the PalmaDNS framework that analyses and determins if a TTL value is suspicious or not.

Namespaces

- ttl

7.10 Programmering/palmadns/Code/PalmaDNS/Framework/queries.py File Reference

Classes

- class PalmaDNS.Framework.queries.Queries

  This is the class for holding all the Passive DNS detected queries that the framework is to anlyze.

Namespaces

- PalmaDNS.Framework.queries

7.11 Programmering/palmadns/Code/PalmaDNS/Framework/query.py File Reference

Classes

- class PalmaDNS.Framework.query.Query

  Holds the Query() class, see the Query docstring for more information.
- class PalmaDNS.Framework.query.Query.QueryData

  This is a nested class to reduce the amount of code that had to be written.

Namespaces

- PalmaDNS.Framework.query

Index

C Kildekode for web-grensesnitt og andre skripts

C.1 /Web/templates/layout.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>PalmaDNS</title>
  <meta name="description" content="Webinterface for PalmaDNS framework"/>
  <style>
  html,body {height: 100%;}
  #wrap {min-height: 100%;height: auto !important;margin: 0 auto -300px;}
  #push {height: 300px;}
  #footer {height: 300px;background-color: #f5f5f5;}
  </style>
  <link href="{{url_for('static',filename='bs.css')}}" rel="stylesheet">
    <link href="{{url_for('static',filename='fui.css')}}" rel="stylesheet">
</head>
<body>
  <div id="wrap">
    <div class="container">
      <div class="row">
        <h1>PalmaDNS <small>WebUI</small></h1></br>
      </div> <!-- END: Header row -->
      <div class="row">
        {% block body %}{% endblock %}
      </div> <!-- END: Content row -->
    </div> <!-- END: Content container -->
    <div id="push"></div> <!-- Push the footer down -->
  </div> <!-- END: Div wrap -->
  <div id="footer">
    <footer>
      <div class="container">
        <div class="row">
          <div class="col-xs-7">
            <h3 class="footer-title">Awww yeahh!</h3>
            <p>Saa bra at du bruker programmet vaart. Eventuelle feil, mangler, og
                 eller bugs kan du sende til /dev/null hvis ikke annet er gitt
                 ekslisitt beskjed om. Du kan kontakte oss paa /dev/null.<br/></p>
          </div> <!-- END: Footer content column -->
          <div class="col-xs-5">
            <div class="footer-banner">
              <h3 class="footer-title">PalmaDNS</h3>
              <ul>
                <li>Pluginbasert</li>
                <li>Fleksibel</li>
                <li>Fremmtidsrettet</li>
                <li>Utvidbar</li>
                <li>Awesome</li>
              </ul>
            </div> <!-- END: footer-banner -->
          </div> <!-- END: col-xs-5 -->
        </div> <!-- END: Footer row -->
      </div> <!-- END: Footer container -->
    </footer>
  </div> <!-- END: Footer div -->
</body>
</html>
```

C.2 /Web/templates/params_form.html

```
{% extends "layout.html" %}
{% block body %}
<form class="form-inline" method="post" action="/">
  <div class="row">
    <div class="col-xs-3">
      <label for="query">Query</label><br>
      <input type="text" class="form-control" name="query" placeholder="Query">
    </div> <!-- END: Query column -->
    <div class="col-xs-3">
```

```
      <label for="answer">Answer</label><br>
      <input type="text" class="form-control" name="answer" placeholder="Answer">
    </div> <!-- END: Answer column -->
    <div class="col-xs-3">
      <label for="ttl">TTL</label><br>
      <input type="text" class="form-control" name="ttl" placeholder="Time To Live">
    </div> <!-- END: TTL Column -->
  </div> <!-- END: Text fields row -->
  <div class="row">
    <br>
    <div class="col-xs-3">
      <button type="submit" class="btn btn-primary btn-lg">Submit</button>
    </div> <!-- END: Button column -->
  </div> <!-- END: Submit row -->
</form>
{% endblock %}
```

C.3 /Web/templates/results.html

```
{% extends "layout.html" %}
{% block body %}
<div class="row">
  <div class="col-xs-3">
    <a href="/" class="btn btn-block btn-lg btn-primary">Tilbake</a>
  </div> <!-- END: Back button column -->
  <div class="col-xs-6"></div> <!-- Empty column spacer -->
  {% if time %}
  <div class="col-xs-3">
    <div class="small">Analyse tok {{ time }} sekunder.</div>
  </div> <!-- END: Timer column -->
  {% endif %}
</div> <!-- END: Header row -->
</br>
<div class="row">
  <div class="col-xs-12">
    <table class="table">
      <tr>
        <th>Client</th>
        <th>Query</th>
        <th>Answer</th>
        <th>Plugin</th>
        <th>Score</th>
      </tr>
      {% for result in results %}
      <tr>
        {% for field in result.split(',') %}
        <td>{{ field }}</td>
        {% endfor %}
      </tr>
      {% endfor %}
    </table>
  </div> <!-- END: Results column -->
</div> <!-- END: Results content row -->
{% endblock %}
```

C.4 /Web/web_ui.py

```
#!/usr/bin/env python

"""This is a prototype for a web user interface for the PalmaDNS pdns
analysis framework. The application is built on the python module Flask.
"""

# Import the needed Flask components to make this app work
from flask import Flask
from flask import request
from flask import render_template

# Import time (for timing of analysis) and sys (for path appends)
import time
import sys

# Import the PalmaDNS Framework Core module
sys.path.append('..')
from PalmaDNS.Framework import core

# The "app" that this page is and will show
app = Flask(__name__)

```

```
# Define a route to the URL '/' with GET and POST methods
@app.route('/', methods=['GET', 'POST'])
def params():

    # If the user hos posted something (HTTP 'POST')
    if request.method == 'POST':
        # Start the timer and create a palmadns instance
        start_time = time.clock()
        palma = core.Core()
        palma.set_output_destination(core.DEST_RETURN)

        # Extract data from the POSTed data
        query = request.form['query']
        answer = request.form['answer']
        ttl = request.form['ttl']

        # Import the POSTed data to the framework and run the analysis
        palma.import_single(query, answer, ttl)
        results = palma.run()

        # Get the results from the analysis and calculate time used
        data = results['data']
        diff_time = time.clock() - start_time

        # Render the results template with the data from analysis and time used
        return render_template('results.html', results=data, time=diff_time)

    # If the request is HTTP 'GET'
    else:
        return render_template('params_form.html')

# This runs when the this file 'web_ui.py' is executed
if __name__ == '__main__':
    app.run()
```

C.5 /large_file_analysis.sh

```
#!/bin/bash

# If the first argument is a file, then asume that it is a pdns-log file
# This is just a PoC so no further checking of data validity is performed
if [ -e $1 ]; then

  # Create a directory to place the temporary files and the results
  mkdir tmp
  cd tmp
  split -l 100000 -a 3 ../$1 tmp_
  mkdir results

  # Some metadata about the number of lines and number of split files
  LINES=$(cat ../$1 | wc -l | awk '{print $1}')
  FILES="$(ls -1 tmp_* | wc -l | awk '{print $1}')"

  # Give the user som fancy output about the analysis starting
  echo -e "\n+-------------------------------------------------------+"
  echo -e "|\tLarge file analysis for PalmaDNS\t\t|"
  echo -e "+-------------------------------------------------------+\n"
  echo -e "\t- Will analyze $LINES lines of pDNS data"
  echo -e "\t- Split into $FILES file(s)"
      echo -e "\t- Start: $(date +"%H:%M:%S")\n"

  # Loop through each split file and run a palmadns analysis on it
  for f in $(ls tmp_*); do
    echo -en "\t+ Started analysis of $f..."
    python ../PalmaDNS run file -i $f -o "results/$f.results"
    echo -e "  [ DONE ]"
  done

  # Tell the user that the analysis is finnished and cleanup the data
  # directories so that the results are in the place the user started
  echo -e "\n\t- Finnished: $(date +"%H:%M:%S")"
  mv results ..
  cd ..
  rm -r tmp/
  echo -e "\t- Results are in ./results/tmp_*.results\n"
fi
```

D Forprosjekt

Prosjektplan

Forprosjekt for bacheloroppgave

Av Lars Christian Andersen og Andreas Moe

1 Mål og Rammer

1.1 Bakgrunn

Mnemonic AS er i dag nordens største leverandør av IT- og informasjonssikkerhet. En av de primære tjenestene de leverer er sikkerhetsovervåkning av kundenes nettverk. Et av mange verktøy som brukes her er datakilden Passive Domain Name System (pDNS). Dette brukes både som kilde for oppdagelse, eskalering, og videre kartlegging av sikkerhetshendelser. Mnemonic er via denne oppgaven ute etter et samlet rammeverk for analyse av sikkerhetshendelser med bruk av deres pDNS database. Dette rammeverket skal basere seg på en kartlegging av dagens teknikker som skadevare benytter for kommunikasjon på internett som kan oppdages og analyseres ved hjelp av pDNS. Fokuset vil ligge i kartleggingen av teknikker brukt av skadevare samt utarbeidelsen av en kravspesifikasjon til rammeverket, hvor en såkalt proof-of-concept skal utarbeides mot slutten av prosjektets gang.

1.2 Prosjektmål

Etter dette prosjektet skal prosjektgruppen sitte igjen med:

- Oversikt over et utvalg teknikker brukt av skadevare hvor man kan dra nytte av pDNS for detektering og eller analyse.
- Vurdering av behovet for et rammeverk basert på vurderte teknikker brukt av skadevare for detektering og eller analyse.
- Utvikle et proof-of-concept av rammeverket som tar for seg et eller flere av de fremlagte teknikkene.
- Fullverdig bachelorrapport med prosess, funn og resultater.

1.3 Rammer

Prosjektgruppen består av to studenter, Andreas Moe og Lars Christian Andersen. Ettersom emnet IMT3912 - Bacheloroppgave IMT er 20 studiepoeng har vi estimert 27.5 til 30 timer i uken per gruppemedlem. Med oppstart beregnet i uke to og at prosjektet presenteres i uke 23 vil dette gi oss 21 uker med arbeid. Totalt vil dette da føre til at gruppen vil kunne tilby rundt 1150 til 1250 arbeidstimer til dette prosjektet. Dette estimatet inkluderer forprosjekt, møter, planlegging, rapportskriving, forskning og muntlig presentasjon.

Prosjektgruppen forplikter å forholde seg til frister satt av Høgskolen i Gjøvik. Dette kan være frister som omhandler innlevering av prosjektplan og innlevering av sluttrapporten. Per den 20/01/2014 er følgende frister gjeldende som viktige rammer for prosjektet vårt: 27/01/2014 - Innlevering av prosjektplan, 19/05/2014 - Innlevering av endelig rapport, 03-05/06/2014 muntlig presentasjon av oppgaven.

Det vil ikke være noen obligatoriske økonomiske eller ressursmessige krav satt fra gruppen sin side. Vi har ingen krav til kontor / arbeidsrom, økonomisk budsjett ut over personlige utgifter eller datakraft og eller annen IT-infrastruktur. Dette gjør gruppens arbeid til svært mobilt og fleksibelt.

2 Omfang

2.1 Fagområde

I dette prosjektet så er gruppemedlemmene inne på en rekke disipliner innenfor fagområdet informasjonssikkerhet. Hovedtemaet for dette prosjektet er teknikker som skadevare bruker for kommunikasjon med kommando- og kontrollservere. Oppgaven fokuserer på de tilfeller hvor denne kommunikasjonen går over DNS, og dermed hvordan disse teknikkene kan detekteres ved hjelp av analyse av pDNS datasett. Medlemmene vil også være innom temaer som systematisk analyse, kartlegging, systemutvikling og reverse engineering.

2.2 Avgrensning

- Prosjektet kommer ikke til å dekke over *alle* teknikker som skadevarer bruker.
- Det resulterende programmets hensikt er kun for å gi et eksempel på hvordan informasjonen opparbeidet i analysen skal brukes.
- Det er kun teknikker fra kjent og publisert skadevare som skal brukes som grunnlag i analysen.

2.3 Oppgavebeskrivelse

Passive DNS (pDNS) er en kilde som kan brukes til å oppdage sikkerhetshendelser og for å utarbeide tidslinjer til videre etterforskning. pDNS har fått annerkjennelse i de senere årene for å være en viktig kilde til analyse av nettverkskommunikasjon og sporbarhet ved sikkerhetshendelser. En utfordring ved slik analyser er at trusselaktører i stadig økende grad blir flinkere til å utnytte den dynamiske naturen til DNS som for eksempel sett ved fast flux, typo squatting og Domain Generation Algorithms (DGA). Da denne typen teknikker har ulike karakteristikker og i variert grad kan identifiseres av andre datakilder, ønsker Mnemonic at teknikkene kartlegges og at det utvikles et rammeverk som støtter opp under utfordringen.

I dag samler Mnemonic inn pDNS data fra prober over hele verden og lagrer dette i en sentralisert database. For å bedre imøtekomme utfordringen ønsker Mnemonic et samlet rammeverk basert på en kartlegging av dagens teknikker, og som over tid kan utgjøre en dynamisk og fleksibel datakilde til deres analytikere.

Oppgaven skal se nærmere på følgende:

1. Kartlegge teknikker som brukes av skadevare og trussel aktører hvor man kan dra nytte av pDNS som en datakilde og vurdere kobling mot domeneregistrarer ved disse teknikkene.
2. Identifisere behovet til et rammeverk basert på vurderte teknikker slik at analytikere får nødvendig informasjon om hendelsen.
3. Utvikle en Proof-of-Concept av rammeverket som tar for seg en eller flere av kartlagte teknikkene.

3 Prosjektorganisering

3.1 Ansvarsforhold og roller

I dette prosjektet har vi to gruppemedlemmer og flere andre involverte parter. Oppdragsgiveren vår er Mnemonic AS, med Andreas Bråthen som kontaktperson. Som veileder har vi Erik Hjelmås som er førsteamanuensis ved Høgskolen i Gjøvik. Gruppen vår består av Lars Christian Andersen og Andreas Moe. Begge er studenter ved HiG på studiet Bachelor i Informasjonssikkerhet.

Som struktur for gruppearbeid har vi valgt en fleksibel og flat struktur. Hovedtanken, som også gjelder for flere aspekter av prosjektet vårt, er å skape en solid grunnmur når det gjelder reglement, organisering og struktur, men fleksibel nok til å fange opp og kunne forbedre og utnytte effektive løsninger som kan dukke opp underveis i prosjektet. Alt dette på en kontrollert måte. Vi velger å ikke utpeke en bestemt gruppeleder ettersom forskjellige faser av prosjektet og forskjellige utfordringer vil kreve at den med mest ekspertise trer frem i en posisjon av bestemmelse. Denne fordelingen av ansvar og "lederløs" struktur er noe som vi begge føler vil fungere bra, og er godt tilpasset vår personlige dynamikk. Ved de fleste store prosjekter ved Høgskolen i Gjøvik iløpet av vår tid her har vi vært på gruppe sammen, men denne erfaringen har vi god tillit til at denne strukturen vil passe oss bra. For å ikke gjøre gruppestrukturen vår til et ukontrollert miljø har vi satt klare rammer når det gjelder arbeidstid, dager vi skal jobbe sammen på, samt ansvarsfordeling. Forskjellige områder av prosjektet vil ha konkret bestemte roller og tildeling av ansvar, mer informasjon om dette kan finnes i vedlegg A - Gruppekontrakt.

3.2 Rutiner og regler i gruppa

Se vedlegg A - Gruppekontrakt

4 Planlegging, oppfølging og rapportering

4.1 Hovedinndeling av prosjektet

Det primære sluttproduktet til dette prosjektet er den akademiske rapporten som skal omfavne prosessen, resultatene og arbeidet utført underveis. Med bakgrunn i dette har vi fra starten av valgt å dele prosjektet organisatorisk i to deler. Den ene delen vil være arbeidet knyttet opp mot rapportskrivingen. Den andre delen er selve gjennomføringen av arbeidet som er kartleggingen, analysen og oppgavens praktiske del. Med grunnlag i dette har oppgavens praktiske del blitt delt opp i tre hovedfaser; kartlegging, identifisering og kravspesifisering, og prototyping.

Kartleggingsfasen vil inneholde generell fordypning i temaet og analyse av større datamengder. Det er estimert at denne fasen skal være ca 50 prosent av prosjektarbeidet. Planlegging og forberedelse er viktig, og det er med dette i tankene at gruppen har valgt å bruke så mye ressurser på denne fasen. Uten et godt grunnlag vil det være vanskeligere, og ikke minst mindre effektivt, å gå videre til de fasene som bygger på den første fasen. Den neste fasen som er identifisering og kravspesifikasjon vil anvende den informasjonen som analysen i den første

fasen har resultert i. Denne fasen er estimert å være ca 40 prosent av prosjektarbeidet. Den siste fasen vil være å utvikle et enkelt rammeverk ut ifra kravspesifikasjonen utarbeided i fase to. Det er ikke så veldig stort fokus på denne fasen og det beregnes at dette skal være ca 10 prosent av prosjektarbeidet.

Da de to første fasene inneholder elementer hvor det er vanskelig å beregne nødvendig ressurser ønsker gruppen å ha en dynamisk prosess hvor det er mulig å gjøre endringer uten at det skal ha store konsekvenser. Gruppen tenker å benytte seg av konseptet Sprint som er brukt i utviklingsmodellen Scrum. Ved å ha delmål i en backlog og trekke ut enkelte før hver sprint får gruppen en mulighet til å dele opp prosessen i mindre delprosesser og dermed gjøre fremgang mer målbart. Gruppen får også mulighet til å velge hvilket områder vi skal fokusere på i den neste sprinten. I den siste fasen ønsker gruppen å benytte seg av en iterativ fremgangsmåte da rammeverket vil være modulbasert og ganske lite. En iterativ modell passer godt med størrelsen på det forventede programmet. Vi føler at selv om Scrum brukes i utviklingssammenheng så kan vi ved å plukke ut enkeltelementer fra modellen lage et system som passer bra for forskning og analyse.

Den andre hoveddelen av prosjektet vil være dokumentasjon- og rapportskriving. Her er ønsket å parallellisere hoveddelene slik at vi får en effektiv prosess hvor dokumentasjon er en sentral del. Ved å følge denne tankegangen vil vi også oppnå en større oversikt over fremgang og har mulighet til å unngå overraskelser når det nærmer seg slutten av prosjektet. Dokumentasjonen vil inneholde kravspesifikasjon og tilhørende informasjon som er nødvendig for å utvikle et rammeverk. Rapporten vil inneholde dokumentasjon over prosessen, funn og resultater.

4.2 Plan for statusmøter og beslutningspunkter

For å opprettholde effektivitet og sikre riktig fremgang ønsker vi å ha statusmøter hver mandag klokken 13:30 til 14:00. Dette er noe vi har kommet frem til i dialog med vår veileder.

Disse statusmøtene skal inneholde grafisk illustrasjon over hvilken fase gruppen er i, og dette skal presenteres i et Gantt-diagram. De skal også inneholde større oppgaver som akkurat er gjennomført og oppgavene planlagt de neste 14 dagene samt oversikt over hvilke av disse oppgavene som eventuelt er underveis og eller ferdigstilte. Passerte milepæler skal også bli diskutert. Kommentarer fra forrige statusmøte skal bli diskutert og valg tatt ut ifra dette skal bli dokumentert. Valg av større viktighet som er tatt siden sist møte skal bli begrunnet kort.

Da denne prosjektgruppen har en flat struktur med en flytende lederrolle kan det oppstå diskusjoner hvor uenighet fører til ingen bestemte valg. I dette tilfellet skal beslutningsprosessen ha en formell form hvor begge medlemmer skal begrunne sitt valg for den andre. Hvis det, etter en slik prosess, ikke blir enighet skal det trekkes om hvilket valg som skal tas.

Grunnen til at denne litt annerledes fremgangsmåten brukes er at begge medlemmer har jobbet godt med hverandre før og kjenner hverandres kunnskapsnivå. Dette, og et felles ønske om et godt resultat, fører til at det legges tillit i at de valgene som fremmes av et av

gruppemedlemmene er gode. Alle statusmøter skal dokumenteres i form av en formell logg, og alle større valg skal dokumenteres med en kort begrunnelse. I de tilfeller hvor det har oppstått en uenighet skal begge meninger dokumenteres uansett hvilket valg som taes.

I starten av hver sprint skal det leveres en statusrapport til oppdragsgiver og veileder. Denne rapporten skal dokumentere fremdrift og eventuelle avvik fra planlagt fremdrift.

5 Organisering av kvalitetssikring

5.1 Risikoanalyse

For å bedre håndtere uregelmessigheter og uønskede hendelser i et hvilket som helst prosjekt er det viktig å gjennomføre en initiell risikoanalyse. Målet med å gjennomføre denne er å få frem de mest fremtredende risikoene som prosjektet er utsatt for, og vurdere konsekvens og sannsynlighet av disse. Til slutt oppsummerer vi med en liste over avbøtende tiltak på de uønskede hendelser med høyest estimert risiko. I denne analysen har vi hentet mange konsepter fra Nasjonal Sikkerhetsmyndighet sin veiledning i Risiko- og sårbarhetsanalyse fra 2004. Vi definerer heretter risiko som: Risiko = konsekvens ganger sannsynlighet. Videre vil følgende presenteres: En liste over forskjellige uønskede hendelser, definisjoner av forskjellige grader konsekvens og sannsynlighet, en risikovurdering samt resulterende tiltak.

ID	Beskrivelse av uønskede hendelser
1	Oppgaven vi har tatt på oss er for vanskelig for oss å beherske på en måte som gir en go sluttrapport i form av akademisk kvalitet. Det kan også hende at oppgaven (p vanskelighetsgrad) må omformuleres nevneverdig.
2	Mnemonic AS som oppdragsgiver bestemmer seg for å ikke gi oss tilgang til nødvend datasett (innhøstet passive dns data).
3	Veileder og/eller kontaktperson hos oppdragsgiver viser liten grad av tilgjengelighet og/e samarbeidsvilje ovenfor prosjektgruppen.
4	Det oppstår uenigheter mellom prosjektgruppens medlemmer som fører til dårlig samarbe og lav grad av fremgang.
5	Et av prosjektgruppens medlemmer møter (gjentatte ganger) ikke til avtalt tid og/eller leve ikke avtalt arbeid til avtalt tid.
6	Det oppstår alvorlig sykdom som gjør et av prosjektgruppens medlemmer utilgjengelige i lengre tidsperiode (her settes 2 uker som en lengre tidsperiode)
7	Det oppstår tekniske problemer med en av lagringsplatformene vi benytter som fører til t av data og/eller manglende tilgjengelighet over en lengre tidsperiode.
8	Det resulterende arbeidet av prosjektgruppens arbeid viser seg å ha liten nyttegrad grunn funn av allerede lignende og/eller bedre løsninger.
9	Prosjektgruppen viser manglende evner til å planlegge og distribuere tiden satt av oppgaven.
10	Et av prosjektgruppens medlemmer blir kastet ut av gruppen, noe som resulterer i bare medlem igjen til å gjøre det resterende arbeidet.

Betegnelse	Beskrivelse
Katastrofalt (K-4)	Prosjektet må avsluttes og/eller kan ikke gjennomføres som planlagt.
Kritisk (K-3)	Prosjektet er i fare for å ikke kan bli gjennomført som planlagt.
Farlig (K-2)	Prosjektets estimerte fremdrift vil oppleve større forskyvninger
Lite farlig (K-1)	Prosjektets estimerte fremdrift vil bli utsatt for mindre endringer.
Svært sannsynlig (S-4)	Hendelsen vil forekomme ofte (mer enn en gang i måneden).
Meget sannsynlig (S-3	Hendelsen vil forekomme flere ganger (en gang i måneden).
Sannsynlig (S-2)	Hendelsen vil forekomme i løpet av prosjektet.
Lite sannsynlig (S-1)	Hendelsen kan forekomme i løpet av prosjektet.

Ut ifra definisjonen av konsekvens og sannsynlighet ovenfor, sammen med våre uønskede hendelser, er det nå mulig å definere en kalkulert risiko for hver av disse. Konsekvens og sannsynlighet er hver rangert i en S/K-1 til S/K-4 hvor 4 er de mest betydningsfulle. En mer visuell representasjon av den kalkulerte risikoen presenteres i matriseformat under listebeskrivelsen.

ID	K	S	R	Forklaring
1	4	1	4	Det er lite sannsynlig at vi blir tillatt å ta i mot en oppgave som er utenfor vå akademiske og faglige nivå. Prosjektgruppens medlemmer er både målrettede o kan vise til tidligere gode prestasjoner i emner som går inn på denne oppgaven.
2	2	1	2	Mnemonic (via kontaktperson Andreas Bråthen) som oppdragsgiver har ve nåværende tidspunkt bare gitt gode tilbakemeldinger og vist sin fulle støtte prosjektet.
3	2	2	4	Det kan alltids hende at veileder / kontaktperson har mye å gjøre og velger nedprioritere oss. Dette kan enten skje pga manglende kommunikasjon eller so et bevisst valg fra deres side.
4	3	3	9	Det er stor sannsynlighet for at prosjektgruppen vil gå på “en smell” her og de Alle gruppers dynamikk er forskjellig og vi kan ikke se bort ifra at uenighe oppstår og samarbeidsevne faller til tider i prosjektet.
5	2	2	4	Det vil forekomme at arbeid eller oppmøte ikke blir gjennomført som avta årsaker til dette kan være forskjellige, men av tidligere erfaringer vil dette skje.
6	3	1	3	Vi kan ikke være helt beskyttet mot sykdom eller uhell, det er sånt som skjer.
7	3	1	3	Når man ikke har kontroll over infrastrukturen kan man ikke gi noen egne garant og tiltak for at tilgjengelighet blir ivaretatt, men igjen, tjenester som google dri kan vise til god statistikk med tanke på tilgjengelighet.
8	1	1	1	Selv om dette blir oppdaget iløpet av prosjektetsfremgang vil nok estime prosjektplan og målsetting forsette som før.
9	3	3	9	Det er en kunst å estimere med riktig utfall.
10	4	1	4	Med tanke på gruppens sammensetning er dette lite sannsynlig med mulighete for forandring må alltid taes med.

	K-1	K-2	K-3	K-4
S-4				
S-3			4, 9	
S-2		3, 5		
S-1	8	2	6, 7	1, 10

I matrisen over ser vi en representasjon av den kalkulerte risikoen med konsekvens på x-aksen og sannsynlighet på y-aksen. Det er primært 4 uønskede hendelser vi ønsker å trekke frem som vi føler har for høy risiko til at vi kan akseptere dem som de er i dag. Dette gjelder uønskedehendelser nummer 3,4,5 og 9. Følgende tiltak taes da med videre i planlegningen:

ID	Tilltak
3	Få regulert hvor ofte statusmøter med kontaktperson og/eller veileder skal avholdes. S strenge interne rutiner for opprettholdelse av disse møtetidene, herunder også utsendelse møteinnkallelse og -referat. Utarbeid alltid agenda for møte på forhånd og hold en god konsis dialog med kontaktperson og veileder under hele prosjektets gang.
4	Sørg for å utarbeide og implementere en oversiktlig måte for rettferdig fordeling arbeidskapasitet, ikke bare etter ønske men også rettferdighet, behov og individu kunnskap. Gruppen må opprettholde regelmessig dialog om fremgang, utfordringer arbeidsfordeling. Dette må reguleres i en gruppekontrakt.
5	Henger mye sammen med tiltaket nevnt for hendelse #4 ovenfor. I tillegg må loggføring arbeidstimer og arbeidsmengde (via f.eks loggbok) reguleres i en gruppekontrakt. Dette g at alle i prosjektet kan se andres og egen innsats samt bli stilt til ansvar for manglend arbeid utført.
9	Det må ikke taes lett på arbeidet med prosjektplanen. Ekstern ekspertise på teamer so prosjektplanlegning og estimering bør konsulteres før ferdig utkast leveres. Det må og legges arbeid inn i å utarbeide en prosjektplan som er fleksibel nok til å takle uforutse hendelser og tilpasse seg til disse.

6 Plan for gjennomføring

6.1 Gantt-skjema

Se vedlegg C - Gantt-skjema for mer informasjon.

6.2 Kommentarer til gantt-skjema

Oppsummert så er gantt-skjema delt opp i tre deler, “Prosjektplan”, “Hoveddel” og “Avsluttning”. Det er i hoveddelen hvor alt arbeidet skjer. Her er prosjektet delt opp i to hoveddeler - arbeid med kartlegging, analyse og kravspesifisering og rapportskriving - hvor de for det meste vil gå parallelt. I hoveddelen er det også en overliggende Scrum prosess hvor alt av arbeidet er fordelt inn i 2 ukers Sprints. Alle arbeidsoppgaver som eventuelt overlapper gjør dette i minst 2 uker for å ikke forstyrre Scrum syklusene.

Vedlegg A

1 Regler

- Hver arbeidsuke skal hvert gruppemedlem arbeide 27,5-30 timer med oppgaven
- All effektiv arbeidstid skal loggføres
- Arbeid med oppgaven skal begrenses så godt som mulig til mellom 0900 til 1700, mandag til fredag. Helgearbeid unngåes såfremt det er mulig for å gi gruppemedlemmene nok fri og restitusjon.
- Faste arbeidstider med prosjektet er mandag, tirsdag og onsdag 0900 til 1700.
- Alt fravær og situasjoner hvor et gruppemedlem vil være utilgjengelig innenfor normal arbeidstid som er tidligere avtalt skal meldes ifra tidligst mulig samt ha en god begrunnelse.
- Alle eventuelle kostnader som gruppen ved full enighet blir enige om å påta seg deles likt mellom alle gruppens medlemmer. Personlige utgifter som f.eks togreiser eller overtidsmat må betales av den enkelte.
- Google Documents / Drive og andre eventuelle løsninger som inneholder dokumenter knyttet til prosjektet skal bli tatt backup av hver mandag og lagres på hver av gruppemedlemenes personlige datamaskiner, samt et annet sted (skyløsning som ikke er Google Drive eller USB-stick).

2 Sanksjoner og Håndtering

- Ved tre brudd på avtaleregler samles gruppen snarest for å diskutere situasjonen.
- Hvis situasjonen ikke forbedres etter dette kontaktes veileder innen en uke for å avklare videre fremgang.

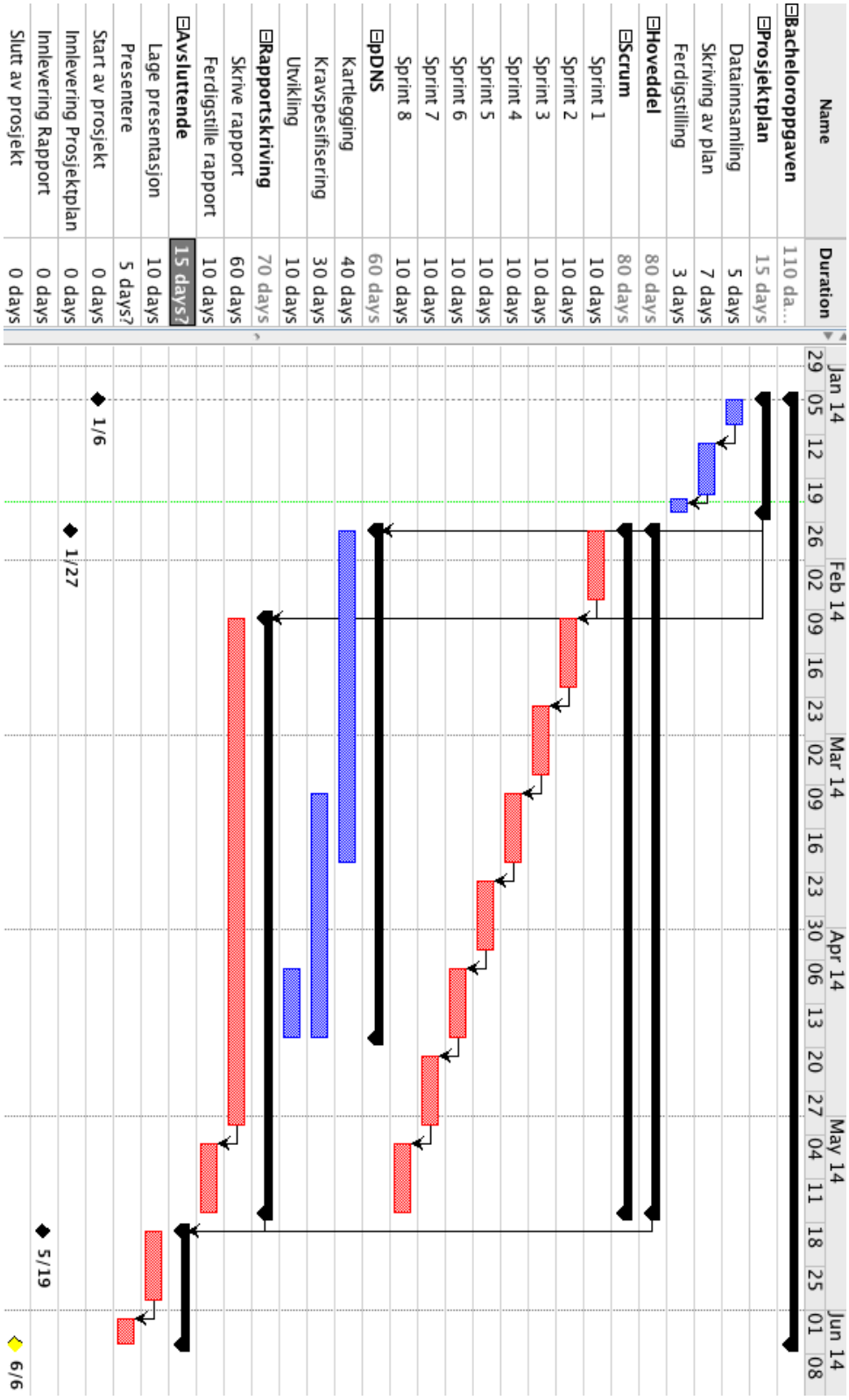
3 Roller og ansvarsfordeling

- Lars har ansvar for oppfølging, drift og administrering av følgende:
    - Loggbok
    - Backup
    - Utsendelse av møtedokumenter (innkallelse, referat, ol.)
- Andreas har ansvar for oppfølgning, drift og administrering av følgende:
    - Subversion repository / LaTeX håndtering
    - Prosjektets Hjemmeside
    - Google Documents / Drive

Vedlegg B

1. Prosjektet startes ved å opprette en komplett liste over oppgaver som må gjøres fremover. Siden det blir en ganske dynamisk prosess vil det bli lagt til elementer i denne listen etter som prosjektgruppen ser nye oppgaver som må gjennomføres
2. En sprint varer i 14 dager og ved starten av hver sprint er det faste elementer som skal gjennomføres:
    a. Et møte med veileder hvor fremgang og videre oppgaver diskuteres.
    b. Et Sprint Planning Meeting hvor vi:
        i. Velger ut oppgaver fra backlog som er aktuelle for den kommende sprinten
        ii. Estimerer arbeidsmengde og viktighet av disse oppgavene
        iii. Velger ut oppgaver som passer inn med tanke på tidsperioden vi har til rådighet i sprinten
        iv. Fordeler oppgavene innad i prosjektgruppen ut ifra tidsbruk og kompetanse
        v. Starter sprinten formelt
3. I løpet av sprinten skal det:
    a. Holdes daglige møter hvor det diskuterer status og eventuelle problemer, og enkel tilbakemelding gis på ferdige oppgaver.
4. Når er sprint er ferdig skal:
    a. Uferdige oppgaver legges tilbake i backlog til neste sprint
    b. Et møte hvor det diskuteres fremgang, avvik og eventuelle oppståtte problemer gjennomføres. Her skal man også se på ting lært og hva som kan gjøres bedre ved neste sprint.

Vedlegg C - Ganttskjema



E Fremgangslogg

Dato	Beskrivelse
09/01-14	Mappestruktur opprettet i Google Docs. Verktøy for arbeid har blitt valgt. Google Docs for samskriving og Redbooth for oppgavestyring.
13/01-14	Første skypekonferanse gjennomført. Tatt valg vedrørende prosjektplan og generelt hvilken retning prosjektet skulle ta. Offisiell start av arbeid på prosjektplan
20/01-14	Hjemmeside opprettet. Utkast til Gantt-skjema generert.
22/01-14	Første møte med veileder.
23/01-14	Prosjektplan ferdigstilt og levert. Offisiell start av arbeid på rapport.
31/01-14	Møte med oppdragsgiver. Mange avklaringer rundt oppgaven og arbeidet videre.
04/02-14	Texpad Connect valg som samskrivingsverktøy på rapporten. Latex-dokument opprettet og mal levert av HiG tilpasset bachelorgruppen.
14/02-14	Dokumentstruktur på rapport bestemt.
19/02-14	Fungerende utkast for Innledning og Kravspesifikasjon
01/03-14	Større endringer i struktur i hoveddelen av rapporten etter nye funn og tilbakemeldinger. Første utkast av rapport sent til veileder og oppdragsgiver.
03/03-14	Start på utvikling av prototype.
11/03-14	Gjennomført møte med arbeidsgiver. Tilbakemelding førte til endringer i prototypekode og rapport.
28/03-14	Fungerende prototype. Flyttet fokus over til rapport.
30/03-14	Problemer med Textpad Connect og gruppen opplevde tap av data. Rapport flyttet over til SVN for videre arbeid.
11/04-14	Nytt utkast sent til veileder.
14/04-14	Dokumentasjon for prototypen ble generert.
21/04-14	Omfattende oppgaveliste for de to kommende sprintene er laget, og gruppen har nå et fullstendig overblikk over oppgaver som må gjøres før prosjektslutt.
29/04-14	Kort møte med oppdragsgiver. Prototype ble presentert og kort tilbakemelding ble gitt.
02/05-14	Stopp for nye elementer i rapporten. Heretter er det kun fiksing av eksisterende innhold som skal gjøres. Utkast levert til veileder for tilbakemelding.
09/05-14	Siste tilbakemelding fra veileder. Innspurt er nå igang.
16/5-14	Bacheloroppgave levert inn.

F Statusrapporter

Statusrapport

Sprint 1

Gjennomførte oppgaver

Vi har endelig kommet igang med arbeided etter at prosjektplanen har blitt lagt bak oss. Vi føler at vi har hatt en fin fremgang iløpet av den første sprinten og satser på at det holder seg slik videre. Etter et møte med Andreas Bråthen fikk vi avklart forskjellige oppfatninger og avgrenset oppgaven nok til at vi begynner å se et vagt lys i tunnelen. Skisse for arkitektur og hendelsesflyt ble laget med tilleggsdetaljene vi fikk fra møtet.

Fremgangsmessig ligger vi godt ann. De planlagte oppgavene for sprint 1 ble gjennomført tidlig og gruppen valgte å legge nye oppgaver inn i den eksisterende sprinten for å holde prosessen gående. Herunder inngår oppsett av Latex og disposisjon av rapport. Da disposisjonen begynner å ta form har gruppen startet å skrive enkelte avsnitt i rapporten.

Avvik

De eneste avvikene fra planen i denne sprinten var at vi ikke hadde satt opp nok oppgaver i sprinten og måtte legge inn nye underveis. Dette kan taes til ettertanke ved videre sprinter.

Neste steg

De to hovedområdene som kommer til å ha mest fokus i sprint 2 vil være kapitlene Introduksjon og Kravspesifikasjon. Vi ønsker å få dette ned tidlig slik at vi har noe fast å skrive rundt. Ved neste sprintmøte skal dette deles inn i mindre oppgaver som kan legges inn i oppgavebehandlingssystemet vårt.

Statusrapport

Sprint 2

Gjennomførte oppgaver

Vi har fått skrevet og revidert kapitlene innledning og kravspesifikasjon/design.

Avvik

Det var et område som ikke ble ferdig denne sprinten. Arbeidet med attributer, listing og sammendrag av disse var en større jobb en planlagt og vil dermed bli ført videre inn i neste sprint.

Neste steg

I denne kommende sprinten vil vi fokusere på delen av rapporten vår som omhandler attributter og tekikker mtp deteksjon og analyse vha pdns.

Statusrapport

Sprint 3

Gjennomførte oppgaver

Kapittelet om Kravspesifikasjon og design er nesten ferdig. 14 attributer har blitt valgt til videre analyse. Etter møte med oppdragsgiver så ble det valgt ut 5 fokusområder av disse 14. Kravspesifikasjon og design og prototype må endres endel.

Avvik

Igjen så er det kapittelet om attributer som ikke er helt fullført.

Neste steg

I denne kommende sprinten vil vi fokusere på endringer på noen kapitler etter ønske fra oppdragsgiver. Deretter vil de 5 attributene bli undersøkt nærmere parallelt med at arbeid på prototype startes.

Statusrapport

Sprint 4

Gjennomførte oppgaver

Etter samtale med oppdragsgiver ble det gjort endel en endringer i hele dokumentet. Kravspesifikasjon ble omstrukturert og arbeidet med prototypen blir gjort parallelt. De 5 attributene det fokuseres på begynner å ta form.

Avvik

Arbeidet med prototypen ble som sagt startet tidligere enn planlagt.

Neste steg

I den kommende sprinten vil vi fokusere på å forbedre kravspesifikasjon og skrive kapittelet om prototype. Deretter er planen å gå igjennom en liste over små forbedringer som har blitt satt til side inntil videre.

Statusrapport

Sprint 5

Gjennomførte oppgaver

Koden for prototype er nå ferdig, og vi står igjen med en fungerede prototype. Det har blitt tatt et valg om å sette et bestemt stopp for videre arbeid på koden for at dette ikke skal gå utover ressurser planlagt til rapportskriving. Diagrammer er oppdatert for å gjenspeile det ferdige produktet. Kravspesifikasjon er ferdigstilt. Kapittelet om deteksjon og analyse er så og si ferdig og det har blitt utarbeidet en oversikt over mangler / ting som burde endres og eller skrives i kapittel 1 - 3.

Avvik

Det var planlagt at kapittelet om prototypen skulle vært startet, men dette har blitt utsatt grunnet arbeid på koden.
Det var planlagt at deteksjon og analyse skulle være ferdig, men her ble konklusjonen utsatt til neste sprint.

Neste steg

Kapittelet om prototypen skal startes på. Deteksjon og analyse skal ferdigstilles og det skal jobbes videre med å holde en rød tråd igjennom de innledende kapittelene. Elementer i oversikt over mangler skal også arbeides på.

Statusrapport

Sprint 6

Gjennomførte oppgaver

Kapitlene 2, 3 og 4 er nå fullførte. Nye delkapitler har blitt opprettet under 1 Introduksjon for å få med nødvendig informasjon om rapporten. Videre har det blitt gjort endringer i disposisjonen på de siste kapitlene. Dette har blitt delt inn i Videre arbeid, Diskusjon og Konklusjon. Det har blitt skrevet mye denne sprinten da vi fikk noe negativ, men konstruktiv tilbakemelding ved forrige utkast. Det har blant annet vært stort fokus på å holde en rød tråd igjennom hele rapporten.

Avvik

Oppgavemessig så opplevde gruppen ingen avvik. Denne sprinten gikk derimot over påsken og gruppen valgte å ta et kontrollert avbrekk fra bacheloren for å samle krefter før siste innspurt. Dette har dermed ført til et noe lavere antall arbeidstimer denne perioden.

Neste steg

Som den siste sprinten med skikkelig rapportskriving har det blitt satt opp en ganske omfattende liste over deler som skal skrives. Mange av disse punktene er emner som allerede er så og si ferdig. Arbeidet her vil kun være å få det over i rapporten på en god og presenterbar måte.

- 1.12 - Terminologi
- 1.13 - Praktisk informasjon
- 5.3.1 - Brukerinteraksjon
- 5.3.2 - Dypdykk i moduler
- 5.3.3 - Resultater
- 5.4 - Plugins
- 5.5 - Bruk av rammeverket i andre programmer
- 5.6 - Konklusjon og drøfting av prototype
- 6 - Videre arbeid
- 7 - Diskusjon
- 8 - Konklusjon
- Skrive innledning og mål / hvorfor
- Forklare bash scriptet
- Forklare flask web-ui
- Skjema - Sammendrag av rapport
- Forord

Disse oppgavene vil bli delt opp mellom gruppens medlemmer og en intensiv periode er i vente.

Statusrapport

Sprint 7

Gjennomførte oppgaver

På fredag 02 mai ble et nytt utkast levert til veileder. Gruppen anser nå rapportens struktur og innhold til å være så og si fastsatt, og det er kun mindre endringer som er planlagt. Alle de planlagte oppgavene ble gjennomført i forrige sprint, og arbeidsoppgaver for kommende sprint har blitt utarbeidet.

Avvik

Ingen avvik denne sprinten.

Neste steg

I den kommende sprinten vil det kun være arbeid vedrørende retting av eksisterende arbeid. Herunder ligger feil ved syntaks, semantikk og logikk. Det skal også jobbes mye med å skape en rød tråd igjennom hele rapporten. Presentasjon har blitt diskutert og vil jobbes med hvis det viser seg at arbeidsmengden i denne sprinten ikke blir så stor.

Statusrapport

Sprint 8

Gjennomførte oppgaver

De to siste ukene har blitt brukt til finpuss av eksisterende produkt. Her har gruppens medlemmer jobbet på kryss og tvers for å være sikker på at alt ble rettet ihvertfall tre ganger. Rapporten anses nå som et komplett produkt og gruppen ser seg fornøyd med det vi har oppnådd det siste semesteret.

Avvik

Det var ingen avvik denne sprinten.

Neste steg

Rapporten leveres idag, 16 mai, inn til sensor. De kommende dagene vil brukes til å koble av og jobbe med eventuelle eksamener. Deretter begynner arbeidet med å forberede seg til presentasjon av rapporten.

G Prosjektavtale

HØGSKOLEN I GJØVIK

PROSJEKTAVTALE

mellom Høgskolen i Gjøvik (HiG) (utdanningsinstitusjon),

Mnemonic as

(oppdragsgiver), og

ANDREAS MOE

LARS CHRISTIAN ANDERSEN (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 01.01.2014 til 01.08.2014.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der HiG yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra HiG å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra HiG. Studentene dekker utgifter for trykking og ferdigstillelse av den skriftlige besvarelsen vedrørende prosjektet.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. HiG står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av faglærer/veileder og sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode, disketter, taper mv. som inngår som del av eller vedlegg til besvarelsen, gis det en kopi av til HiG, som vederlagsfritt kan benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av HiG til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved HiG og/eller studenter har interesser.

   Besvarelser med karakter C eller bedre registreres og plasseres i skolens bibliotek. Det legges også ut en elektronisk prosjektbesvarelse uten vedlegg på bibliotekets del av skolens internett-sider. Dette avhenger av at studentene skriver under på en egen avtale hvor de gir biblioteket tillatelse til at deres hovedprosjekt blir gjort tilgjengelig i papir og nettutgave (jfr. Lov om opphavsrett). Oppdragsgiver og veileder godtar slik

AB

offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og dekan om de i løpet av prosjektet endrer syn på slik offentliggjøring.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i Fronter. I tillegg leveres et eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med et eksemplar til hver av partene. På vegne av HiG er det dekan/prodekan som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og HiG som nærmere regulerer forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene.

   Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale, skjer dette uten HiG som partner.

10. Når HiG også opptrer som oppdragsgiver trer HiG inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene i mellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

HiGs veileder (navn): Erik Hjelmås

Oppdragsgivers kontaktperson (navn): ANDREAS BRÅTHEN

Student(er) (signatur): _____ dato 17.01.2014

_____ dato 22.01.2014

_____ dato _____

_____ dato _____

Oppdragsgiver (signatur): _____ dato 11.01.2014

IMT Dekan/prodekan (signatur): _____ dato 13.02.2014

Versjon Januar 2011hb