BACHELOROPPGAVE:

BO22EB-02: CYBER GATHERING & DETECTION TOOL

Kay Siver Bø

29. mai. 2022

Dokumentkontroll

Rapportens tittel: BO22EB-02: Cyber Gathering & Detection Tool	*Dato/Versjon* 29. mai. 2022/0.3.2
	Rapportnummer: B022EB-02
Forfattar(ar): Kay Siver Bø	*Studieretning:* AUT
	Sidetal m/vedlegg 25
Høgskolens rettleiar: Tom Kjøde, tom.kjode@hvl.no	*Gradering:* Open
Eventuelle Merknader: Forfattar(ane) tillét at oppgåva kan publiserast.	

Oppdragsgivar: ABB AS	*Oppdragsgivars referanse:*
Oppdragsgivars kontaktperson(ar) (inklusiv kontaktinformasjon): *Erik Serck-Hanssen, erik.serck-hanssen@no.abb.com*	

Revisjon	**Dato**	**Status**	**Utført av**
0.0.1	13.05.2022	Vidareføring av forstudie.	Kay Siver Bø
0.1.0	16.05.2022	Utkast. Nye overskrifter og nytt appendiks.	Kay Siver Bø
0.2.0	22.05.2022	Nye overskifter. Automatisk generert ordliste.	Kay Siver Bø
0.2.1	23.05.2022	Meir brødtekst.	Kay Siver Bø
0.3.0	28.05.2022	«SNAFU»	Kay Siver Bø
0.3.1	29.05.2022	Utkast til godkjenning	Kay Siver Bø
0.3.2	29.05.2022	Figuroppdatering	Kay Siver Bø

Forord

Takk til Erik Serck-Hanssen som bidrog med rettleiing og tok seg tid til å ordne med nødvendige systemtilganger, og Anders Hole og Emin Vagap som har stått for oppfølging av oppgåva og svart på førespurnader og spørsmål rundt klokka. Utan dykk ville prosjektet stagnert.

Vidare må eg takke Tom Kjøde for sin urokkelege støtte som rettleiar. Utan din innsikt og erfaring ville ikkje oppgåva ha oppnådd sin noverande form.

Til slutt må eg heidre dei som har ofra både tid og livsglede for å kunne kvalitetssikre oppgåva.

Tusen takk, alle saman.

Samandrag

Programsystemet for rapportgenerasjon møtte ikkje dei opprinnelege måla satt for prosjektet.

Dei mest kritiske manglane omhandlar støtte for McAfee-programvare, og primitive rapportar. Eksperimentell støtte for Acronis- og vCenter-programvare er implementert, men ikkje testa. Bruk i produksjonsmiljø er ikkje anbefalt per dags dato.

Innhald

1 Innleiing

1.1 Om prosjektgruppa

Prosjektgruppa består berre av ein person.

1.2 Oppdragsgivar

1.2.1 ABB

ABB er eit europesisk aksjeselskap som driv hovudsakeleg med sal av elektrisk utstyr. Selskapet vart oppretta i 1988 og kan spore sitt opphav til svenske Allmänna Svenska Elektriska Aktiebolaget, ASEA, og sveitsiske Brown, Boveri & Cie, BBC, og har opp gjennom åra absorbert selskap som Elektrisk Bureau AS [1].

ABB har i 2021 om lag 105 000 tilsette [2].

1.3 Problemstilling

Oppdragsgivar overvakar fleire kritiske produksjonssystem, og lagar vekentlege statusrapportar for desse systema. Desse rapportane blir per dags dato laga manuelt.

Oppgåva blir å utvikle programvare som kan automatisere denne manuelle prosessen, på ein forsvarleg måte.

1.4 Hovudidé for løysingsforslag

Oppdragsgivar vil ha eit program som baserer seg på ein sentralisert (nettverks/program)arkitektur og REST API eksponert av serverprogramvare. Programmet er tiltenkt å køyrast manuelt ein gong i veka for å generere ein statusrapport.

2 Kravspesifikasjon

Frå oppgåveteksten kan ein trekkje ut følgjande nøkkelpunkt om det ynskja produktet[1]:

- *Utvikle ett enkelt system for overvåkning og monitorering av windows server produksjonssystem.*
- *Systemet skal kunne generere rapporter med KPI metrics om produksjonsmiljøet sin tilstand.*
- *Basere seg på scripting for innsamling av data og presentere disse på en god måte gjennom grafisk brukergrensesnitt, eller rapporter*
- *Applikasjonen krever god portabilitet og enkel installasjon (gjerne med bakoverkompabilitet på eldre Windows server versjoner).*
- *Høye krav til oppetid på systemet og at datainnsamling fører til lite påvirkning på produksjonssystem.*
- *Systemet bør ha muligheter for å generere krypterte rapporter for å hindre at viktig systemdata blir tilgjengelig for andre.*
- *Overføre monitoreringsdata vha tcp/ip eller kryptert epost mot incident handling system(f.eks ved alarmer).*
- *Samle logger fra kjente verktøy som f.eks Acronis, McAfee antivirus server, SIEM og VMware.*

Desse nøkkelpunkta kjem frå når oppgåva var berekna på ei større prosjektgruppe. Etter dialog med oppdragsgivar kjem følgjande krav fram:

- Programmet må køyre under Windows Server 2016
  - Utviklast i .NET Framework 4.7.2 (eller eldre) og/eller PowerShell
  - Vere mest mogleg «sjølvstendig» (Døme: ein programfil med alle (køyretids)avhengigheiter inkludert/integrert)
  - Kan ikkje stoppe viktige prosessar som konsekvens av programfeil.
- Programmet må kunne fungere saman med Acronis Backup, McAfee EPO og Vcenter
  - må klare å generere ein rapport med data innhenta frå desse programma.

[1] Utklipp frå oppgåveskildring.

3 Analyse av problemet

Krava slik dei vert presentert av oppdragsgivar er få, og heller absolutte. Programstabilitet, kompatibilitet med Windows Server 2016 og samhandling med programma Acronis Backup, McAfee EPO og vCenter, er krav som må oppfyllast skal programvara utvikla i denne oppgåva vere nyttig. Programvara som skal utviklast kan heller ikkje påverke vertsystema i nokon grad (for å oppretthalde høg «oppetid»), som gjer at resursbruk må vere sterkt avgrensa.

Om ein ser vekk i frå dette er oppgåva eigentleg rimeleg open. Funksjonane til den ynskja programvara kan oppsummerast som

- datainnsamling (primært frå programma Acronis Backup, McAfee EPO og vCenter) og
- rapportgenerering (i eit lett tilgjengeleg filformat).

Dersom ein har tid kan funksjonaliteten til den ynskja programvara utvidast til å bli ein form for rapporterings- og overvakingssystem. Dette inneberer at «programmet» blir køyrd som ein serverprosess, og utvida med ny funksjonalitet som

- krypterte rapporter, og
- automatiske varsel per e-post (i feilsituasjon).

Som konsekvens av redusert resursbruk på serversida, kan ein ikkje installere mykje meir programvare utover det som følgjer med operativsystemet. Dette gjer .NET Framework (versjon 4.7.2 eller eldre) og PowerShell til førsteval som utviklingsplattformar, spesielt då oppdragsgivar har erfaring med desse plattformene. Dimed vil tilgjengelege programmeringsspråk vere

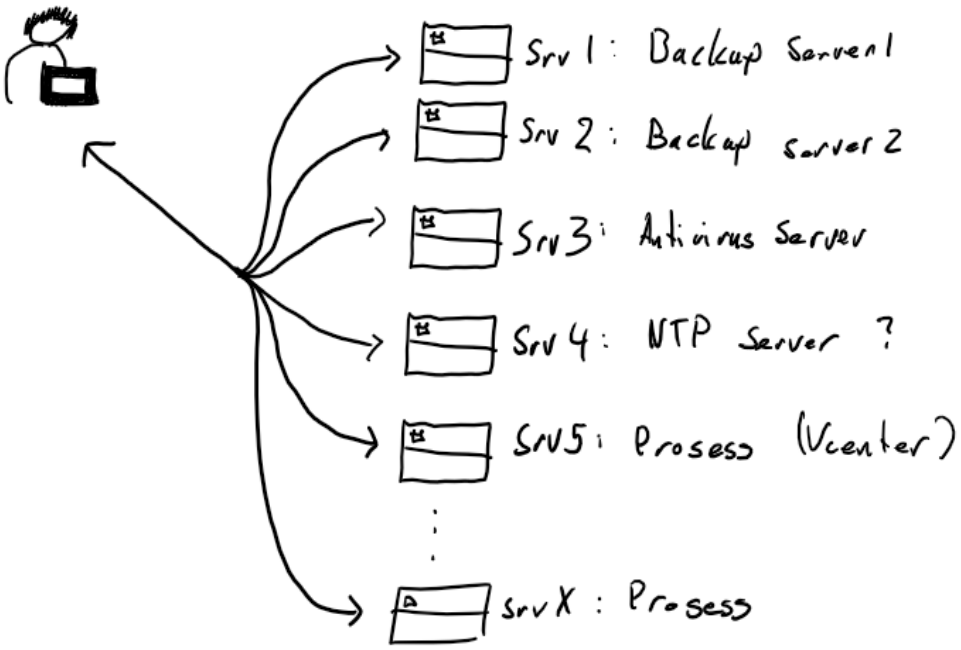
1. C,
2. C++,
3. C#, og
4. PowerShell.

3.1 Utforming av moglege løysingar

3.1.1 Generelt om utforminga

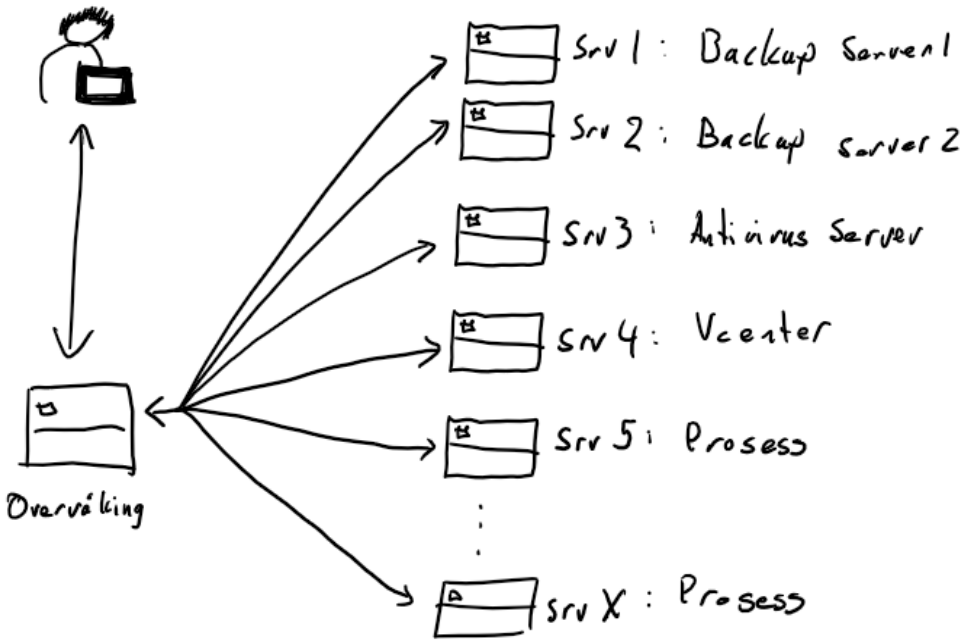
Hovudpoenget med oppgåva kan tolkast til å automatisere ein manuell prosess. Då ein del av denne prosessen framleis kjem til å vere manuell, sjølv etter at oppgåva er fullført, bør ein ikkje gå for langt vekk frå det systemet som er i bruk frå før av, for å gjere overgangen til det nye systemet enklare.

Dimed bør rapportane programmet kjem til å generere, vere rekneark med ein bestemt innhaldsstruktur, då det er dette som er i bruk per dags dato.



Figur 1: Modell for tiltenkt programvarearkitektur

For å gjere den utvikla programvara meir framtidssikkert har prosjektgruppa fatta ei slutning om at alle programbibliotek som kan, skal skrivast/utviklast i .NET Standard 2.0. Dette kjem at .NET som plattform har sett ein del endringar dei siste åra, som gjer det meir attraktivt å utvikle bibliotekskode mot eit statisk kompilasjonsmål støtta fleire eldre versjonar av .NET (og framtidige versjonar) i staden for ein bestemt versjon av .NET.



Figur 2: Modell for programvare med ekspandert funksjonalitet

3.1.2 Implentasjonsvurderingar

3.1.2.1 Datainnsamling

I prinsippet kan den automatiserte datainnsamlingsoppgåva utførast med fleire forskjellige metodar. Her er eit utval over nokre av dei som var vurdert som praktiske til denne oppgåva:

1. Installer eit program på kvar server og bruk dette programmet til å samle data om serverdrift som ein kan hente ut med eit klientprogram.
2. Lag eit (klient)program som kan hente ut eit augeblinksbilete av serverdrifta, og eventuelt aktuelle logger.
3. Lag eit programsystem som med regulære intervall kan hente ut data om serverdrift, og lagre aktuelle data ved behov.

Etter samråd med oppdragsgivar, vart det fatta slutning om at ein hovudsakeleg ynskjer eit programsystem som ikkje er avhengig av installasjon av ekstra programvare på serverane, som heller kan skalerast opp til eit system illustrert i Figur 2. Dette gjer at modellen vist i Figur 1 blir grunnlag for tidleg programutvikling.

3.1.2.2 Rapportering

Rapportering av serverdrift kan skje på fleire måtar,

- E-post, og
- Utskrift av tekst på skjermen til ein «operatør»

er kjente måtar å rapportere på. Eit mogleg problem her er at oppdragsgivar ynskjer at all data skal lagrast i ei reknearksfil. Kjente filformat vil vere

- .ods (ODF)
- .xlsx (OpenXML)
- .csv og .tsv (Tekstbasert).

ODF- og OpenXML-standardane baserer seg på Zip-arkiv med XML «innmat».[2] Dette vil medføre implementasjonskompleksitet om ikkje eksterne programbibliotek blir nytta. Tekstbaserte format er trivielle å programere for, men er sterkt avgrensa i kva innhald dei kan ha, samanlikna med komplekse filformat.

Oppdragsgivar føretrekker OpenXML (.xlsx), men set ingen faste krav til bruk av dette formatet til rapportering, og har også ytra eit ynskje om automatisk sending av rapport over e-post i eit utvida programsystem.

[2] Zip-arkiv med ein bestemt struktur av XML-dokument

4 Implementasjon

4.1 Om programsystemet

I tråd med kravspesifikasjonen vart eit rapportgenerasjonssystem utvikla. Grunna bekymringar rundt kompleksitet, vart konfigurasjon og rapportgenerasjon implementert av prosjektgruppa som separate program. Programma er kalla ReGen og ConfigMaker.

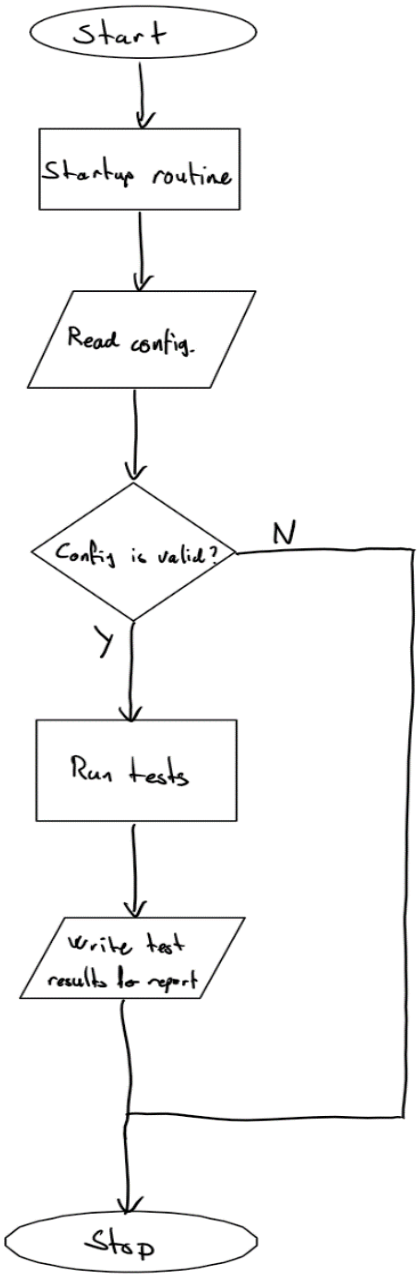
Kjeldekoda til programma er tilgjengeleg på GitHub[3].

4.1.1 ReGen

ReGen (frå «Report Generator») er eit fleirplattform konsollprogram som har ansvar for sjølve rapportgenerasjonen. Rapportfilane er av filtypen XLSX.

Forenkla programoppførsel kan representerast som vist i Figur 3.

Bruk av ReGen er skildra i Appendiks C .



Figur 3: Flytdiagram for ReGen

4.1.1.1 API-klientar

Alle API-klientar nytta internt i ReGen, arvar i frå same superklasse for å forenkle implementasjonen. Tradisjonelt er REST API tilstandslause, men i tilfellet med Acronis og vCenter, treng ein å autentisere seg fyst for å få utlevert ei øktnøkkel som skal nyttast vidare. Øktnøkkelen er kun gyldig i eit bestemt tidsrom, og må opprettast på nytt ved tidsutløp. Øktnøklane gjer dimed at klientane har ein «tilstand», men denne er ikkje teken omsyn til i noverande implementasjon. Det er forventa at ReGen utfører testane sine før nøkkelen utgår.

Skal klientane gjerast meir robuste må ein nytte ein form for tilstandsmaskin inne i klient-klassen som automatisk forhandlar ny nøkkel ved utløp.

NB: Acronis- og vCenter-klientar har ikkje blitt tilstrekkeleg testa, og er ikkje anbefalt til bruk i produksjon.

4.1.1.2 SSH-skript

Bruk av SSH til å køyre skript på ei målmaskin, var meint som eit utvidingsalternativ. Med denne funksjonaliteten skal ein kunne klare å hente data og

[3] Rapporten tek utgangspunkt i commit 81cffaf63cce0d466145ca42f5f1326187137160 på «master»-greina. Kjeldekode: https://github.com/tupubozu/BO22EB-02 [5]

køyre testar/kommandoar. ReGen klarar ikkje å tolke desse resultata på same måten som det blir gjort med REST API.

Skript må inkluderast i konfigurasjonsfila som ein «Job» av kategori «Tilpassa», sjå Appendiks C for detaljar.

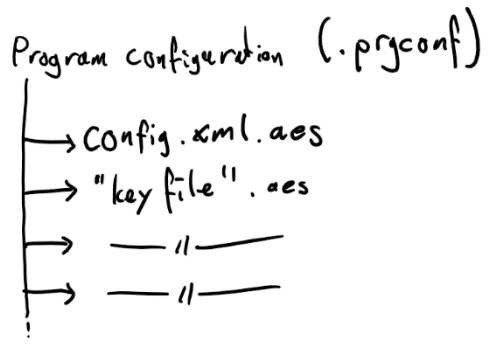
Som eit tryggleikstiltak bør ein ikkje la skripta køyrast av ein høgt privilegert brukar. Ein eigen dedikert brukar med sterkt avgrensa rettigheitar bør opprettast til bruk med ReGen sin SSH-funksjonalitet.

4.1.2 ConfigMaker

ConfigMaker er eit WinForms program (grafisk brukergrensesnitt) som lagar konfigurasjonsfiler som ReGen nyttar til å samle informasjon som skal inkluderast i rapportar.

Bruk av ConfigMaker er skildra i Appendiks C .

4.1.2.1 Konfigurasjonsfil



Figur 4: Konfigurasjonsfiloppbygging. Standardnamn på inkluderte nøkkelfilar er framleis ikkje bestemt. Namn på vedlegg endrar ikkje oppførsel ved lesing av konfigurasjonen så lenge namna er nytta i «config.xml.aes».

Konfigurasjonsfila eksisterte konseptuelt tidleg i planlegginga som ein enkelt XML-fil ReGen skulle lese inn ved programstart. Ein ville at XML-fila skulle kunne lagast manuelt, men etter å jobba med deler av ReGen vart det klart at ein enkelt fil potensielt ville innehalde for mykje informasjon til å handsamast manuelt.

XML-fila kom til å innehalde sensisitiv informasjon, i klartekst, som API- og SSH-nøklar. Dimed ville ein ha XML-fila kryptert. Vidare ville ein ikkje gjere for mykje med SSH-nøkkelfiler, då ein ikkje ynskja å handsame forskjellige teiknsett internt i programmet. Dette medførte at denne filtypen blir lest inn som den er, kryptert, og lagt inn i eit Zip-arkiv saman med ein kryptert utgåve av den opprinnelege XML-fila, som vil innehalde både krypteringsnøkler og initialiseringsvektorer som er unike for kvar vedlagte nøkkelfil.

Krypteringa av konfigurasjonsfila skal fungere som eit vern mot interne og eksterne truslar, som vondsinna brukarar og datainnbrot. Det vil ikkje vere godt nok å ha ein kopi av fila, då ein treng ein nøkkel for lese av innhaldet.

Då konfigurasjonsfila sitt innhald er å rekne som enkelt å gjere ugyldig, er krypteringa hovudsakleg meint å fungere som ei tidkrevjande hindring ein angripar må overkomme.

4.2 Om utviklingsprosessen

Utviklingsprosessen kan skildrast som «forsinka».

Oppgåva vart tildelt bachelorprosjektgruppa seinare enn venta, slik at noko av arbeidet formelt ikkje kunne startast før januar. Normalt forventa gruppa å få tildelt ei oppgåve i slutten av desember. Dette førte til at forstudiet av oppgåva vart gjort hurtig.

Frå fyste møte med oppdragsgivar var tilgangar til testverktøy etterspurd. Enten som tilgang til testsystem, eller som tilgang til testsystemprogramvare. Som konsekvens vart det undersøkt, på oppdragsgivar si side, om prosjektgruppa sitt private utstyr kunne nyttast. Då dette vart avvist vart det sett i gong ein ny prosess, på oppdragsgivar si side, om å få utlevert godkjent utstyr til prosjektgruppa. Utstyret vart utlevert over ein månad etter fyste førespurnad.

Utsetjingane førte til at prosjektgruppa måtte innskrenke omfanget til funksjonsnivået til MVP.

I nedetida har prosjektgruppa fokusert på å granske tilgjengeleg dokumentasjon, og implementere programobjektklasser for API-klienter og konfigurasjonsfilstruktur, og verktøy for konfigurasjonsfilhandsaming (ConfigMaker). Utviklinga skjedde i fyste omgang utan testing, som førte til ein treigare itterasjonssyklus då ein ikkje kunne vere sikker på at nye endringar ville gi ynskja resultat. Til slutt byrja prosjektgruppa å jobbe med SSH-funksjonalitet i rapportverktøyet (ReGen) sjølv om dette var rekna som eit tillegg, fordi det var lettare å teste.

Etter testsystemtilgang var på plass, fullførte ein SSH-funksjonaliteten og gjekk tilbake til utvikling av REST API-klientar og klientintegrasjon i ReGen.

4.3 Implementasjonsvurderingar

4.3.1 Program

4.3.1.1 ReGen

Erfaring med programvare utvikla for GNU/Linux, utgjorde grunnlaget for slutninga om at ein ikkje ville ha eit konsollprogram som kan ta mange argument for å justere oppførselen ved køyretid. Dette gjorde ideen om ei konfigurasjonsfil veldig attraktiv, om ein ville ha noko som skal reknast som brukarvenleg.

Grafisk brukargensesnitt (GUI) vart også vurdert som tiltak for å auke brukarvenlegheita, men grunna manglane informasjon om kva verktøy/plattform oppdragsgivar kom til å nytte i framtida, valde ein å gå fram med konsollprogram grunna breiare kompatibilitet frå .NET plattforma. Ein annan vinkling av vurderinga var at ein ville utvikle noko som kunne vere relevant i ei lengre periode, på bakgrunn av brei kompatibilitet.

Bruk av Serilog-bibliotek vart grunngitt med at ein ville ha eit loggføringssystem integrert i programmet med ein gong, slik at ein lettare kunne utvikle serverprogramvare om ein fann tid til det mot slutten av prosjektet. Prosjektgruppa har også tidlegare erfaring med bruk av Serilog.

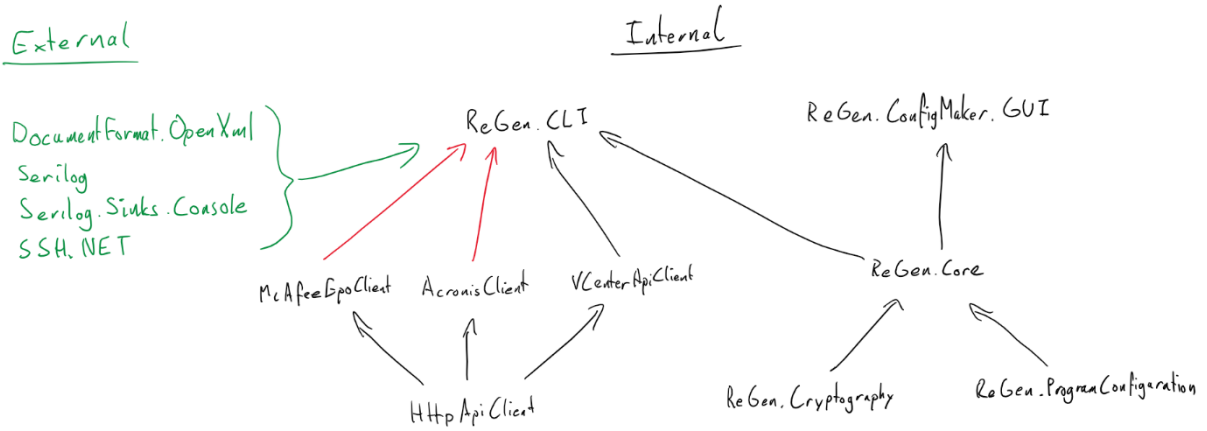
Noverande utgåve av ReGen klarar å generere rapportar basert på SSH-funksonalitet. Rapportane som blir generert er veldig ulike referanserapporten supplert av oppdragsgivar, og grensar til «primitive».

4.3.1.2 ConfigMaker

Etter kvart som at konfigurasjonsfila vart utvikla, såg ein at det vart vanskelegare å handsame fila manuelt. Dette sparka i gang utviklinga av støtteverktøy for konfigurasjonsfilhandsaming.

For ConfigMaker valde ein å nytte grafisk brukargrensesnitt med ein gong då prosjektgruppa sjølv trengde eit lettvint verktøy til å lage testkonfigurasjonar. Gruppa valde å sjå vekk frå breiare systemkompatibilitet til fordel for brukarvenlegheit, då ein i prinsippet ikkje treng å køyre ConfigMaker meir enn ein gong. Det same kan ein ikkje seie om ReGen, som etter problemstillinga er meint til å køyrast rundt ein gong i veka.

4.3.1.3 Programbibliotek



Figur 5: Bibliotekavhengigheiter. «Eksterne» bibliotek vil seie bibliotek utvikla utanfor prosjektgruppa si kontroll. «Interne» bibliotek vil seie bibliotek utvikla av prosjektgruppa til bruk i prosjektet. Raud pil markerer avhengigheit som ikkje er realisert.

Utforminga/organiseringa av programbibliotek vart gjort for å modularisere kjeldekoden. Modulariseringa har fungert godt under utviklinga av ConfigMaker.

Som ein del av API-klientbiblioteka kjem ein del klasser som representerer JSON-dokument ein kan få tilsendt ifrå ein tenar. Skulle ein ha implementert ein fullverdig vCenter-klient ville dette føre til biblioteket ville innehalde fleire klassar enn nødvendig for utføre arbeidet ein treng i ReGen. Ein fullverdig vCenter-klient vil vere så omfattande at den kunne blitt rekna som ei eiga semester-/bacheloroppgåve. På bakgrunn av dette har gruppa berre fokusert på funksjonaliet i klientane som er nyttig for ReGen. Acronis-klienten er mykje enklare, og samstundes meir fullstendig, enn vCenter-klienten i skrivande stund. Ein kan diskutere om ein ikkje burde opprette enda ein superklasse som Acronis-klienten og vCenter-klienten kan arve frå slik at ein kan skille ut felles oppførsel.

4.3.2 Konfigurasjonsfil og filkryptering

Konfigurasjonsfila i sin tidlegaste form skulle vere eit «handskrevet» XML-dokument. Etter kvart som at kompleksiteten auka, måtte ein ty til meir kompliserte løysingar.

Eit problem vart oppdaga under design og implementasjon av SSH-funksjonaliteten i ReGen. SSH vil tradisjonelt krevje enten eit passord, eller ei privat nøkkel, for å autentisere ein brukar. Dette førte til at konfigurasjonsfila måtte innehalde enten eit brukarpassord eller ei privat nøkkel. Under eit postulat om at ei privat nøkkel gjerne har blitt oppretta frå før, gjekk ein vidare med arbeidet for å inkludere den i konfigurasjonsfila. For å sikre portabilitet kan ikkje konfigurasjonsfila innehalde ei sti til fila, men sjølve filinnhaldet.

Inspirasjon vart henta frå EPUB-filformatet, og OpenXML- og ODF-standardane, som førte til at ein baserer konfigurasjonsfila på eit Zip-arkiv, med XML innmat.

Etter at tilgang til testsystema vart innvilga, fatta prosjektgruppa ein slutning om å kryptere innhaldet i Zip-arkivet som eit tryggleikstiltak. Kvar fil i Zip-arkivet er kryptert med AES-256 kvar for seg. I «Debug»-modus blir CBC chiffermodus nytta, og for «Release»-mode blir CFB chiffermodus nytta.[4]

[4] Bruk av CFB og CBC kjem av at CBC vart nytta under store deler av utviklinga og ein ville behalde kompatibilitet med eksisterande testkonfigurasjonar, samstundes som ein vil ha tryggleik i eit «utgitt» program. [4]

Hovud-XML-fila i Zip-arkivet er alltid det siste som blir kryptert, då denne fila inneheld dei unike nøklane og startverdiane nytta av AES-algoritmen under kryptering av vedlagte filer. Grunna XML si regulære natur er hovud-XML-fila i Zip-arkivet det svakaste leddet i heile konfigurasjonsfila sitt kryptografiske oppsett. Konfigurasjonsfila burde dimed ikkje reknast som ei «trygg» lagringsstad i seg sjølv. Krytpteringa av Zip-arkivinnhaldet bør reknast som ei tidkrevjande hindring.

4.3.3 Nettverkskommunikasjon

All nettverkskommunikasjon mot REST API skal vere hardkoda til å nytte HTTPS då vanleg HTTP ikkje er trygg nok. HTTP vil sende data som klartekst. Om ikkje tenaren er sett opp til å støtte HTTPS, vil ReGen feile. Nettverksport for HTTPS-trafikk er valfri.

SSH-kommunikasjon følgjer standard oppførsel på valfri nettverksport.

4.4 Avvik frå kravspesifikasjon

Kravet om utvikling i .NET Framework 4.7.2 har ein gått bort ifrå, då det oppstod problem ved køyring av ConfigMaker på eit tilgjengeleg testsystem. I staden har ein gått over til .NET 6, då dette rammeverket gir betre fridom til å kompilere uavhengige programfiler. Program utvikla i .NET Framework vil krevje at eit køyretidsmiljø er installert på maskina der dei skal køyre. Med .NET 6 vil denne avhengigheita kunne ignorerast.

Samhandling med McAfee EPO blir nedprioritert. Python-bibliotek som gjer samhandling mogleg er distribuert av McAfee, men utan dokumentasjon på dei underliggande REST funksjonane, vil ein krevje SW reverse engineering for å få denne funksjonaliteten med i ReGen. IronPython eksisterer som eit alternativ, men i skrivande stund er IronPython versjon 3 av «beta»-kvalitet og IronPython versjon 2 har veldig få bibliotek som framleis fungerer grunna Python 2-base. [3]

4.5 Gjenståande arbeid

Før at ReGen-systemet kan nyttast i produksjon bør følgjande punkt fullførast:

- Acronis-klienten byrja å bli klar for ein test, men treng nokre linjer med kode i ReGen for å kunne takast i bruk.
- vCenter-klienten må framleis implementerast og testast. I tillegg må ein fullføre kode i ReGen som nyttar klienten til informasjonsuthenting.
- Generert rapport bør validerast inkrementalt, på ein per-modul-basis (SSH, Acronis, vCenter). Innhaldet i rapporten må gjennomgå ein verifikasjonsprosess mot referanserapporten til oppdragsgivar for å sjå om ynskja informasjon blir henta ut.

Andre punkt som bør vurderast fullført er:

- GUI til ConfigMaker har nokre element som burde vere deaktivert i enkelte situasjonar for å hindre brukarfeil.
- McAfee-klienten vil trengje reimplementering i C# om ein skal halde på kravspesifikasjonen. Revurder krava, og vurder eventuelt å lage bindingar for Python-biblioteket.

Referansar

[1] Wikipedia, «ABB,» 15 Desember 2021. [Internett]. Available: https://no.wikipedia.org/wiki/ABB. [Funnet 2 Februar 2022].

[2] Wikipedia, «ABB,» 24 Januar 2022. [Internett]. Available: https://en.wikipedia.org/wiki/ABB. [Funnet 2 Februar 2022].

[3] IronPython, «Release IronPython 3.4.0-beta1 · IronLanguages/ironpython3 · GitHub,» 30 April 2022. [Internett]. Available: https://github.com/IronLanguages/ironpython3/releases/tag/v3.4.0-beta1. [Funnet 23 Mai 2022].

[4] Wikipedia, «Block cipher mode of operation,» 19 Mai 2022. [Internett]. Available: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation. [Funnet 29 Mai 2022].

[5] K. S. Bø, «tupubozu/BO22EB-02,» 24 Mai 2022. [Internett]. Available: https://github.com/tupubozu/BO22EB-02. [Funnet 29 Mai 2022].

Appendiks A Ordliste

A

ABB
Asea Brown Boveri 2; 7

AES
Advanced Encryption Standard 15; 16

API
Application Programming Interface 7; 13; 20

C

CBC
Cipher Block Chaining 15

CFB
Cipher Feedback 15

G

GUI
Graphical User Interface 20

H

HTTP
Hypertext Transfer Protocol 16; 20

HTTPS
Hypertext Transfer Protocol Secure 16; 20

J

JSON
JavaScript Object Notation 15

M

MVP
Minimum viable product 14

O

ODF
Open Document Format 11; 15

R

REST
Representational state transfer 7; 16; 20

S

SSH
Secure Shell 13; 20

SW
Software 16

X

XML
Extensible Markup Language 11; 13

Appendiks B Prosjektleiing og -styring (Forstudie)

B.1 Prosjektorganisasjon

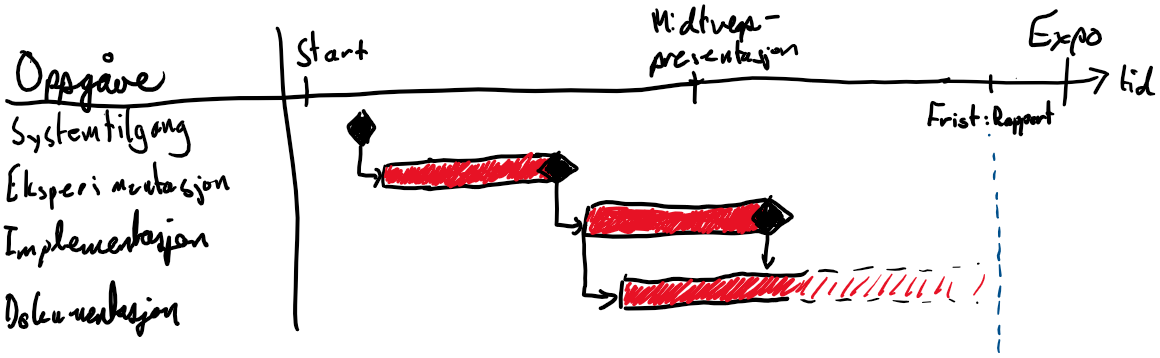
Prosjektgruppa består av ein person. Intern gruppestruktur blir dimed unødvendig.

B.2 Prosjektform

Eigendefinert prosjektform basert på testdreven programvareutvikling(?). Grunna gruppestorleiken vil prosessen/prosjektforma likne på «fossefallsmetoden» dersom observert frå eit tidsperspektiv.

B.3 Framdriftsplan

Grunna svært få gruppemedlem, vil faktisk framdriftsplan vere basert på sjekklister med utgangspunkt i kravspesifikasjonen.



Figur 6: Grunntanke for utviklingsframdrift

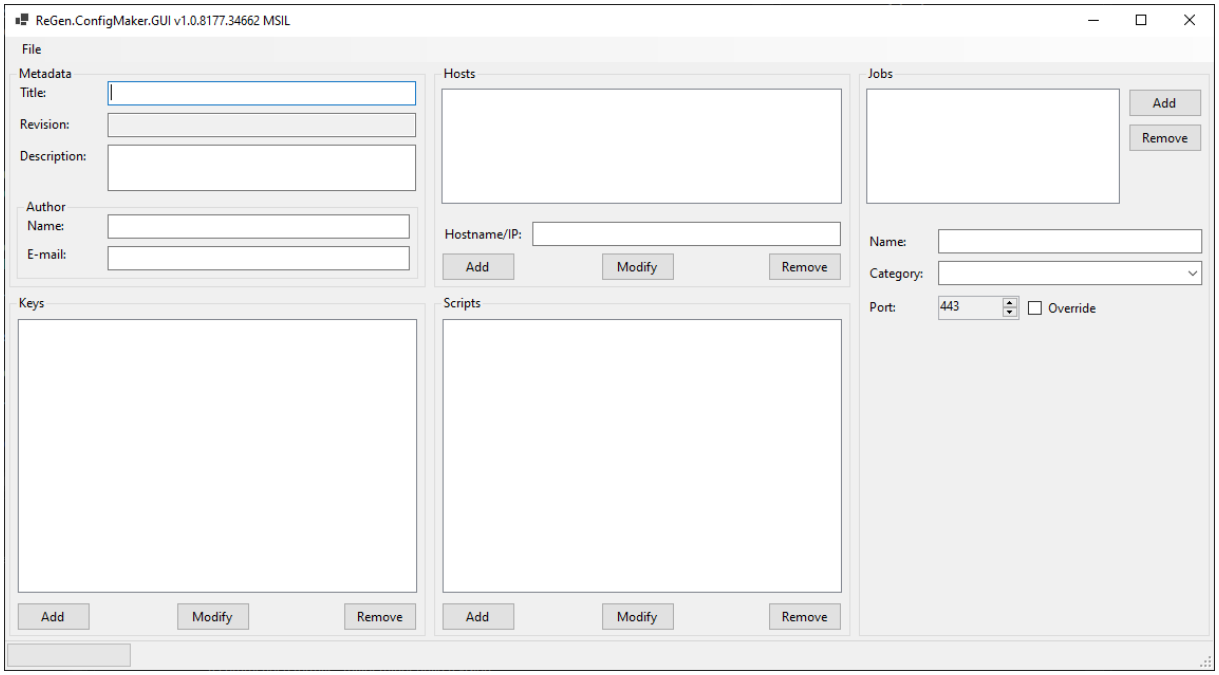
Provisorisk sjekkliste:

- REST API (acronis, mcafee, vcenter) [85 timar]
  - Ping server (HTTP vs HTTPS), valider API.
  - Prøv ikkje destruktiv kommando
  - Hent ut data (Parse data)
- OpenXML [80 timar]:
  - Lag testfil:
    - Eksperimenter med skriving til fil
      - Fleire ark
      - Formlar?
      - Farge?
  - Imiter referanserapport
    - Før inn data
    - Bestem datastruktur/representasjon for API-data
      - Tilpass for skriving til rekneark
- SSH [85 timar]
  - Sjekk plattform = Windows
    - Tidssynkronisering
      - w32tm /query /status
        - w32tm /resync /force
    - Diverse… (valfritt skript?)
- GUI?
  - Web-basert?
- Dokumentasjon/rapport [150 timar, variabel]

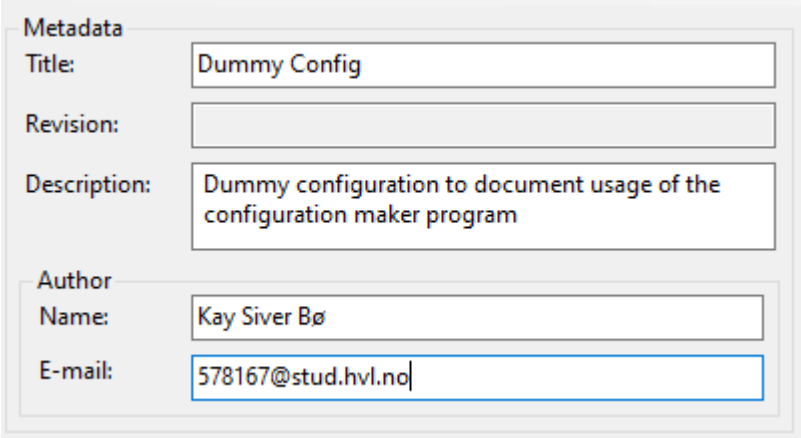
Appendiks C Bruk av ReGen-programsystemet

C.1 Lag ein konfigurasjonsfil (ConfigMaker)

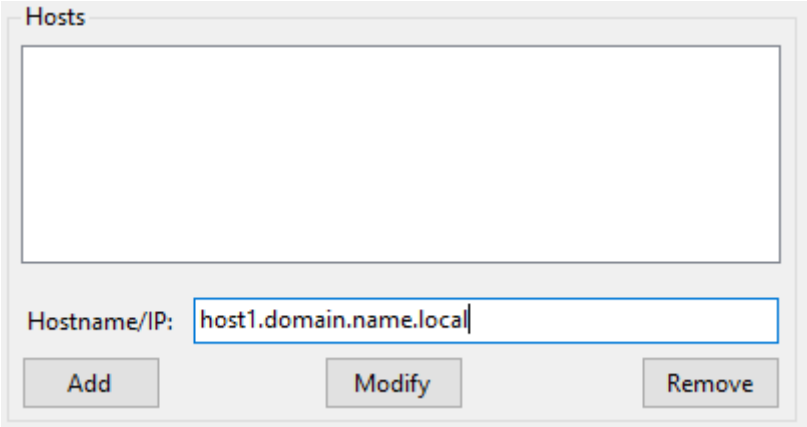
1. Start ConfigMaker.



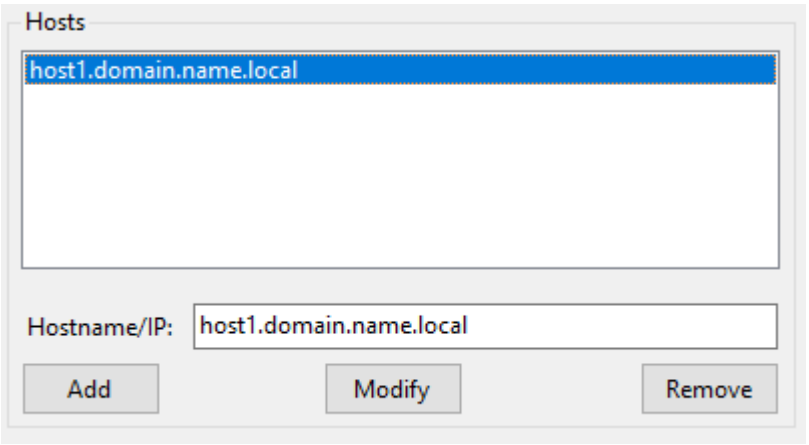
2. Fyll inn informasjon i felta øvst til venstre i vindauget i området kalla «Metadata». Merk at feltet «Revisjon» ikkje skal fullast inn manuelt.



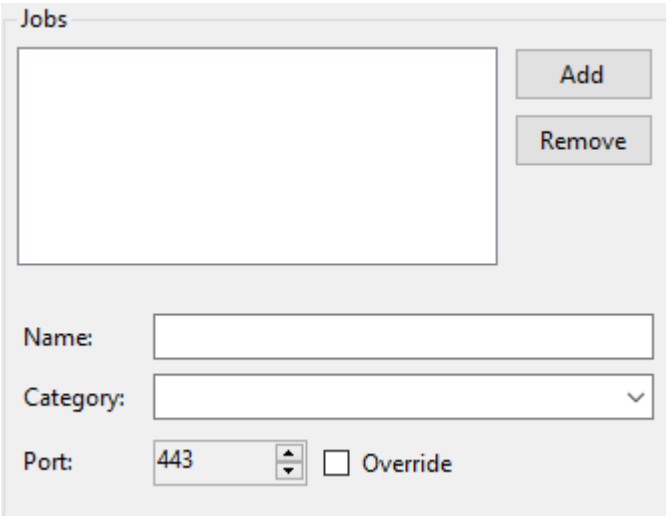
3. Opprett ein «vert» (Host) i ved å skive kvalifisert vertsnamn eller IP-adresse inn i «Vertsnamn/IP» feltet og trykke på «Legg til»-knappen.



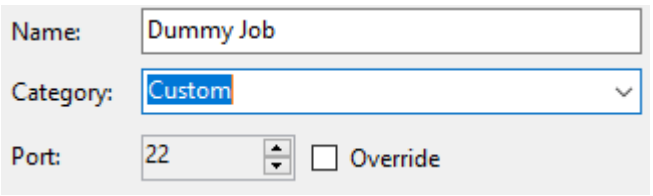
4. Klikk på den nyleg oppretta verten i lista for å markere den. «Jobber» er assosiert med kvar sin vert. Dimed må ein vert vere markert om ein vil leggje til ein «Jobb».



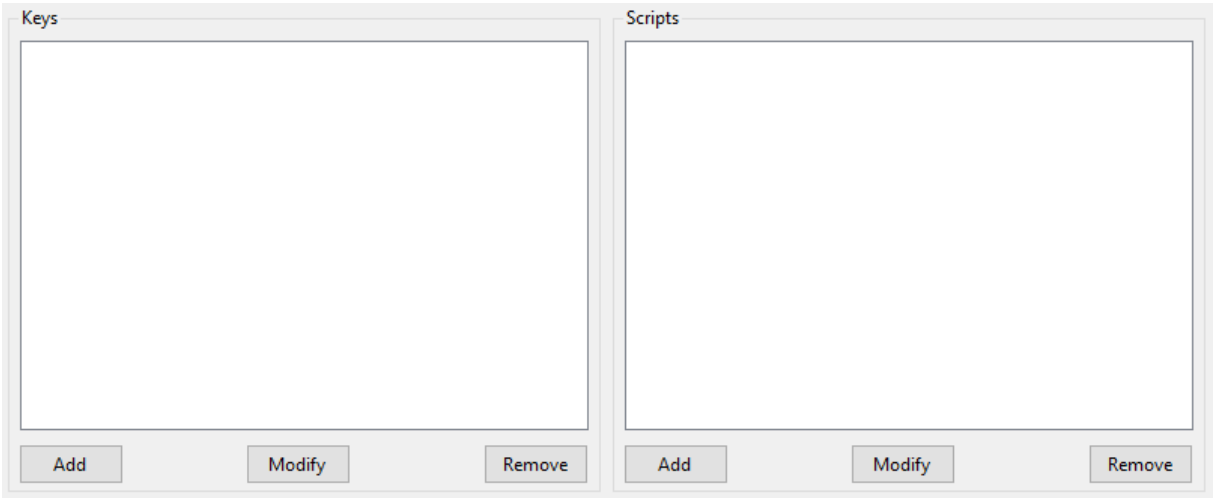
5. Klikk på «Legg til»-knappen under «Jobber».



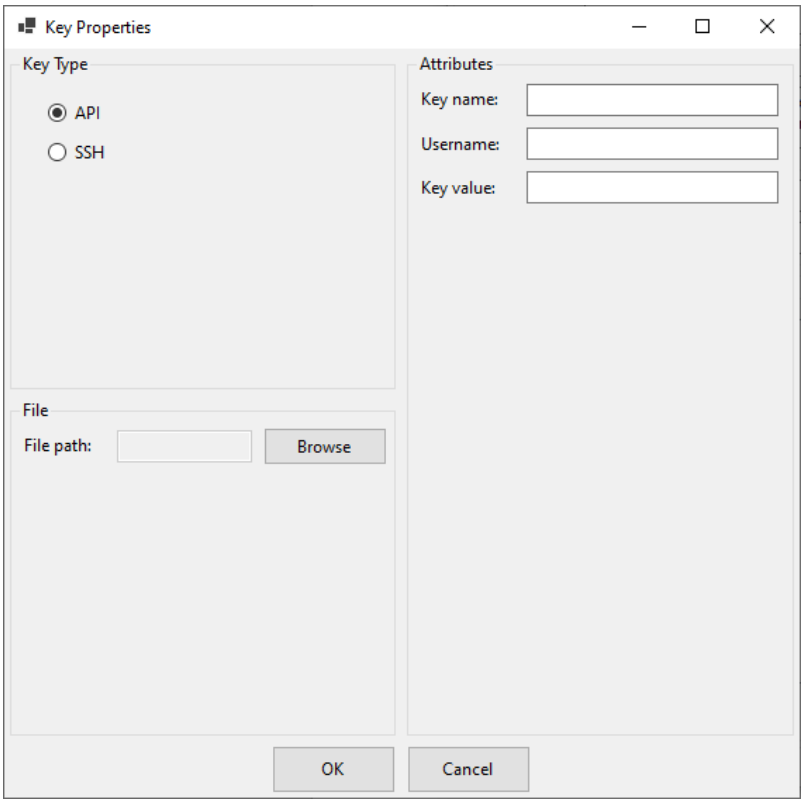
6. Marker den nyoppretta «Jobben». «Namn»-feltet kan fyllast ut med valfritt skildrande oppgåvenamn. Endre «Kategori» og «Port» etter behov. Merk at endring av port krevjer «Overstyring»



7. Legg til «Skript» og «Nøkler» avhengig av kva «Jobbkategori» ein har vald for «Jobben».



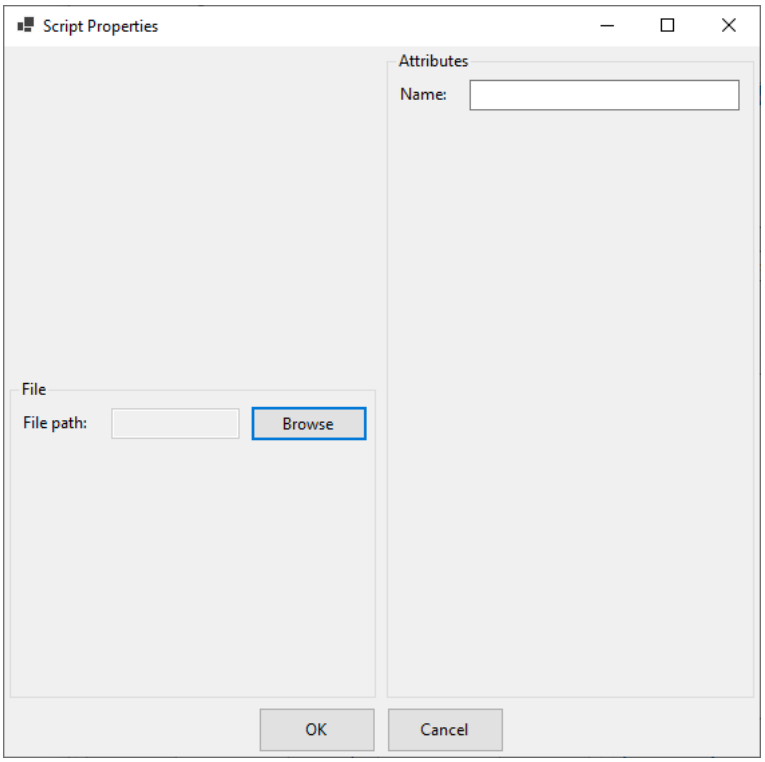
Merk at det kun er «Tilpassa» kategorien som nyttar både «Skript» og «Nøkler». Alle andre kategoriar krevjer kun «Nøklar». Om «Skript» blir lagd til nokon annan kategori enn «Tilpassa», blir dei ignorert av ReGen.



Nøkkelimport blir gjort med å spesifisere nøkkeltype. «API»-nøkkeltypen skal nyttast for alle «Jobb»-kategoriar unntatt «Tilpassa». «Nøkkelnamn»-feltet er eit sjølvvald namn på nøkkelen ein vil leggje til.

For «API»-nøkkeltypen skal kun nøkkelnamn og nøkkelverdi spesifiserast. Alle andre felt vil bli ignorert. Om «Jobb»-kategorien er satt til «Acronis» eller «vCenter» skal nøkkelverdi-feltet fyllast ut på forma «*brukarnamn*:*passord*» der ein nyttar brukarnamn og passord for den korresponderande serverprosessen som køyrer på den spesifiserte «Verten».

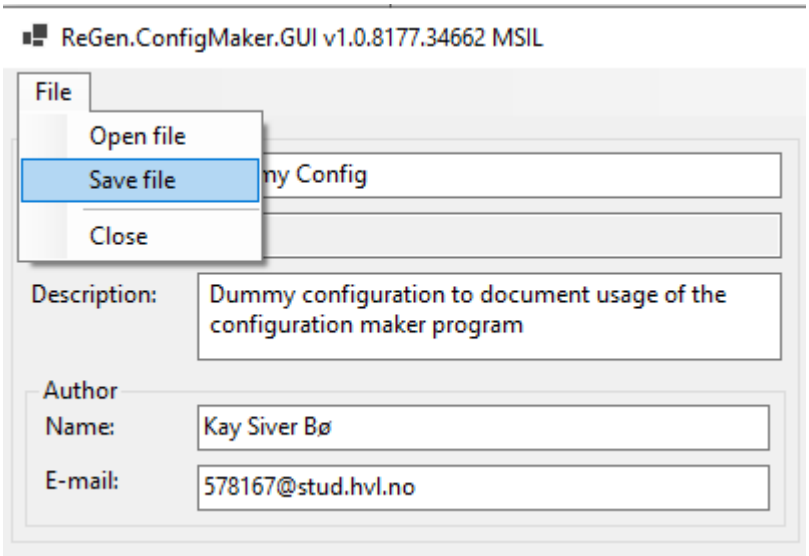
For «SSH»-nøkkeltypen skal nøkkelnamn- og brukarnamnfelta fyllast ut. Brukarnamnet som skal fyllast ut er eit gyldig brukarnamn på «Vert»-maskina. Fila som inneheld privatnøkkelen nytta av SSH til å kople oppgitt «Vert»-brukar må spesifiserast under «Fil». Privatnøkkelfila må vere på eit format støtta av SSH.NET.[5]



Skriptimport blir gjort ved å skive inn eit sjølvvald namn og velje ei skriptfil.

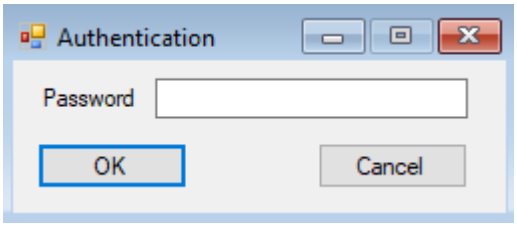
NB: Hugs å trykkje på «OK»-knappen for å fullføre nøkkel- og skriptimport.

8. For å lagre konfugurasjonsfila vel ein «Fil» i menylinja til hovudvindauget, og dertetter «Lagre fil». Spesifiser filnamn og lagringsplass, og trykk «Lagre».



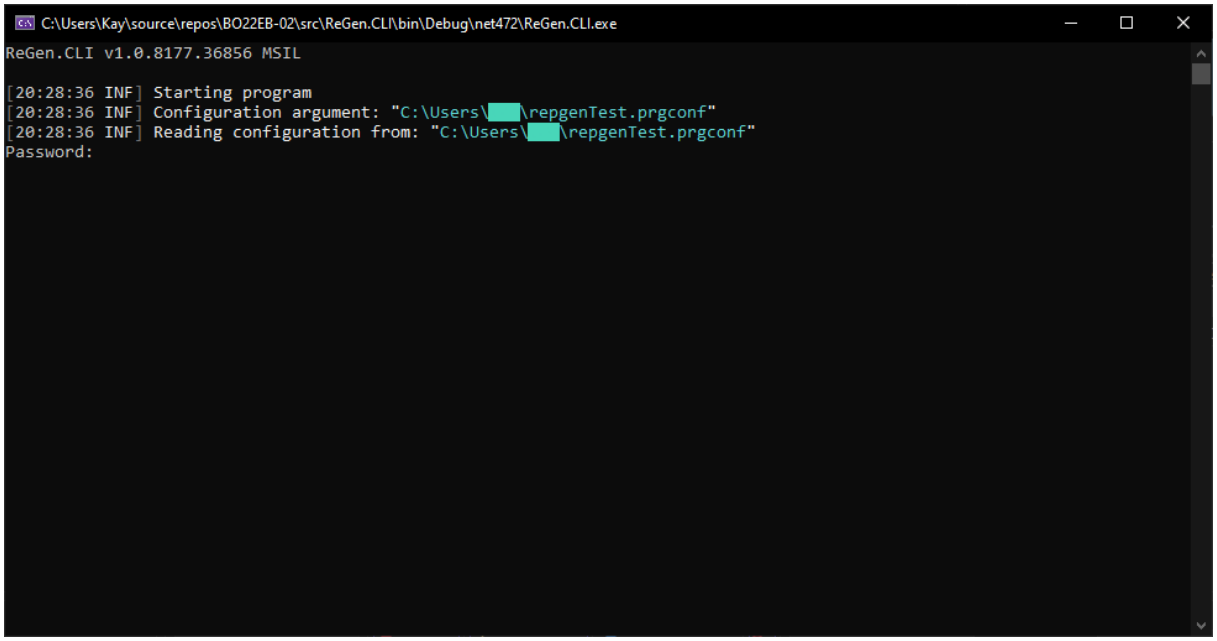
[5] https://github.com/sshnet/SSH.NET/#public-key-authentication

9. Vel eit passord og skriv det inn. Dette blir passordet som ReGen vil spørje etter, og passordet som ConfigMaker vil krevje om ein vel å opne ei konfigurasjonsfil for redigering.



C.2 Køyr ReGen

ReGen startast frå kommandolinje med stien til konfigurasjonsfila som fyste argument.



Skriv deretter inn passordet som vart nytta når ein lagra konfigurasjonsfila. Deretter er det berre å vente til at programmet gjer seg ferdig. Så lenge konfigurasjonsfila er gyldig og ikkje inneheld skript med uendeleg lang køyretid, skal programmet avslutte utan problem.

Dersom ReGen køyrast med konfigurasjonsfiler laga av ConfigMaker fil rapportfila bli lagra som «report.xlsx» i arbeidsmappa.