# Rapport

# Arkitektur- og kildekodegjennomgang EVA Skanning

# Valgdirektoratet

Versjon 1.0
13.6.2018

## 1 Sammendrag

mnemonic har gjennomført en kodegjennomgang av de ulike delene av EVA applikasjonen. Denne rapporten beskriver funn og forbedringsforslag for EVA Skanning. Denne rapporten er delvis basert på manuell kode- og arkitekturgjennomgang og delvis på funn fra analyseverktøyet MicroFocus Fortify SCA, samt vurderinger av hvordan EVA Skanning brukes i dagens løsning, som selvstendige applikasjoner kjørende på eksternt kontrollert utstyr i stemmelokalene.

## 2 Om deloppdraget

For denne komponenten fikk vi i oppstartsmøtet følgende ønske om spissing av oppdraget:

*EVA Skanning 50%*

- *Lokalt installert – ikke kontroll på infrastruktur (sikkerhetsanbefalinger)*
- *Applikasjonssikkerhet*

- *Kodekvalitet*

Vi fikk i tillegg ekstra tid (30 timer) til å se på:

- *Hvordan sikre EVA Skanning applikasjonen best mulig i «ukjent» infrastrukturmiljø*

# 3 Generelt om applikasjonen

Dette kapittelet inneholder observasjoner knyttet til kvalitet og arkitektur. Dette er basert både på vår jobbing med koden og få den til å kompilere og bygge, samt dypere analyser av strukturen og potensielle problemer vi ser med tanke på vedlikehold, videreutvikling og rene sikkerhetsaspekter.

## 3.1 Arkitektur

Vi har få funn som går på arkitekturnivå i EVA Skanning. Vi har dog noen observasjoner som er relaterte.

EVA Skanning bruker stort sett design patterns og rammeverk Microsoft har utviklet. To av disse, Prism og Unity, slapp Microsoft som åpen kildekode i 2015 og overførte ansvaret for videreutvikling til ildsjeler som hadde vært involvert i den tidligere utviklingen. Begge prosjektene har blitt videreutviklet, men når vi søker etter informasjon om Unity vi ser at det er spørsmål om hva som skjer med Unity, som er et rammeverk for Dependency Injection (DI) eller Inversion of Control (IoC). Den ser dog ut til å fortsatt være svært populær, og det gjør at det er flere som potensielt kan ta over utviklingen om den skulle stoppe opp. Det finnes konkurrerende åpen kildekoderammeverk, og noen av disse ser ut til å være minst like aktivt vedlikeholdt, bl.a. AutoFac og SimpleInjector. Microsoft har selv utviklet noe DI-rammeverksfunksjonalitet i .NET Core, dog ser det ut til å være best støtte for ASP.NET i dag. Nå skal et IoC-rammeverk gjøre at klasser skal ha løs knytning mellom seg for enklere å kunne bytte ut disse ved behov. Det er dog neppe like lett å bytte ut IoC-/DI-rammeverk som å bytte ut koden som bruker DI-/IoC-rammeverket. Vi anbefaler dog å se på alternativene, slik at et eventuelt framtidig bytte kan gjøres uten altfor store overraskelser, og da over til noe som har nødvendig funksjonalitet og som blir aktivt vedlikeholdt.

*Anbefaling 1 Skaff oversikt over alternativer til Unity-rammeverket*

Selv om bytte av ReadSoft til et annen OCR-funksjonalitet strengt tatt ikke er en arkitekturendring er det dog en større endring og som vil kunne fjerne noen iboende sårbarheter i dagens løsning.

Valgdirektoratet kjenner svært godt til begrensningen i ReadSoft-løsningen, og ingen av punktene under er nyheter, men det viser at det er store utfordringer ved bruk av ReadSoft-løsningen, og at selv om man vet hva man har skal ikke det være noen sovepute for å finne en ny løsning.

- Den mest alvorlige av disse er at ReadSoft krever Administrator-rettigheter, også for utviklere da Visual Studio må kjøres som Administrator for at byggingen skal fullføre (i alle fall iht den «utgående» dokumentasjonen). Dette er et klart brudd på «Least privilege principle» og det øker muligheten for at ondartet kode får vellykket fotfeste da angrepskoden slipper å bry seg om et normalt vanskelig steg: privilegieeskalering, og som normalt krever en sårbarhet i omliggende infrastruktur, som en Windows-sårbarhet.
- ReadSoft krever også binding mot 32-bits arkitektur og COM-teknologi, som ikke brukes av noen andre komponenter i løsningen. Med mindre det skjer en nyutvikling av ReadSoftapplikasjon/-bibliotek vil den bli mer om mer teknologisk utdatert.
- Bruken av ReadSoft i dagens løsning gjør noen «hacks» for å omgå måten ReadSoft er tenkt å brukes slik at EVA får mer kontroll over hva som presenteres på skjermen. Dette er ikke en direkte sårbarhet, men det viser at det må legges skrives kode som ikke skulle være nødvendig å ha, da et ekte bibliotek vil gi brukeren av biblioteket kontroll.

Nå er det allerede gjort en god del arbeid med å teste ut alternativer til ReadSoft, hvor bl.a. Tesseract, en gammel konkurrent til ReadSoft, og som i dag vedlikeholdes av Google. Tesseract er dog utviklet i C/C++, og det vil kunne være noen begrensninger da det vil være en mer ekstern komponent enn en løsning tilbudt som .NET-bibliotek. Som for mye annen teknologi ser det ut til at det er mye aktivitet på å tilby løsninger i skya, og det kan være at det er en trussel for en å finne vedlikeholdte løsninger for desktop-installasjon da utvikling av skyløsning kan gå på bekostning av desktopløsningen. Dette er noe vi selv ser i verktøyene vi bruker til daglig.

## 3.2 Kodekvalitet og vedlikeholdbarhet

Etter oppgradering til .Net 4.7.1 hvor to passelig store metoder i Valg.EVA.Skanning.Signing.SignatureCryptographicServiceProvider relatert til ble reduserte til én liten metode og fjernet bl.a. bruk av BouncyCastle. Det er kun én BouncyCastle-bruk i igjen i C#-koden. Den ene av de to metoden nevnt over ligger dog fortsatt (ubrukt) i koden, mens den andre er kommentert vekk - inntil den nye er verifisert at fungerer som den skal. I tillegg er nå Valg.EVA.Skanning.Signing.CryptoOids ikke lenger i bruk, og eneste referansen til den er i den kommenterte koden. Denne endringen er separat fra andre endringer etter oppgradering til 4.7.1, og ved å gjøre endringen komplett før innsjekking vil det gi en klasse som kun inneholder aktiv kode. For å få til det må den gamle versjonen av SignData() og tilhørende WritePemSignature() fjernes, samt avhengigheter til eksterne komponenter (som BouncyCastle). Om det skulle vise seg at den nye koden ikke fungerer er det fullt mulig å rulle tilbake endringen da det kun er endringer relatert til denne signeringsfunksjonaliteten som ble endret.

Vi har riktignok ikke sett mangle slike eksempler, men vi vet fra egen erfaring at det er lett å «fjerne» gammel kode ved å kommentere den vekk, og så glemme å rydde senere. Over tid kan det føre det til uoversiktlige klasser, samt avhengigheter til både egen og ekstern funksjonalitet som ikke er i bruk. (Vi fikk f.eks. flere treff med sikkerhetsverktøyene i den ubrukte koden av, inkl. CryptoOids.)

*Anbefaling 2 Bruk mulighetene git gir for å modifisere én og én funksjonsendring*

Som i EVA Admin er det også i Valg.EVA.Skanning.Signing.Certificates en referanse til Scytl, men denne gangen som en streng brukt uttrykk for å sjekke om sertifikatutsteder er en betrodd utsteder:

```
28:         private X509Certificate2Collection
FilterBuyPassCertificates(X509Certificate2Collection certificates) 29:
{
30:             var filterNames = "BuyPass|Scytl|Buypass AS-
983163327".ToLower().Split('|').ToList();
```

Vi ser ingen grunn til at Scytl-sertifikater skal brukes i 2019-valget, og vi antar derfor at dette er en forglemmelse i opprydding fra eVote-tiden. En grunn til at det ikke er så lett å se dette er at teksten som inneholder «Scytl» ligger midt i koden, og er ikke skilt ut til en mer sentral plass for konstanter eller konfigurasjonsdetaljer. Vi har sett at det er en strenger som ligger innbakt i koden og som ikke er plassert på en standardisert plass, som f.eks. i starten av klassen kan tekstkonstanter stå, slik at alle vet hvor de skal lete etter slike når det er behov for å endre en strengkonstant.

*Anbefaling 3 Fjern alle referanser til Scytl*

*Anbefaling 4 Lag et system som gjør at det er lett å finne fram til tekstkonstanter som koden bruker.*

I Valg.EVA.Skanning.Tools.LocalDbToolProxy brukes strengkonstanter for å finne hvilken «Program files»-katalog som et inneholder et angitt binær. Vi skal se at det finnes alternative måter å gjøre det på.

I tillegg er det her for en sjelden gang noe kommentarer før metoden, men kommentarene er ikke oppdatert når koden er det og referansen til SQL Servers filstiversjon 110 (SQL Server 2012), men metoden som returnerer selve stien inneholder katalogen 120 (SQL Server 2014):

```
22:         /// Tool located in
23:         /// C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\
24:         /// or
25:         /// C:\Program Files\Microsoft SQL Server\110\Tools\Binn\
26:         /// </summary>
27:         private static string GetCorrectPathToTool()
28:         {
29:                 string[] programfilesDirectories =
{@"c:\programfiler",@"c:\programfiler (x86)", @"c:\program files", @"c:\program
files (x86)" };
30:          foreach (var programfilesDirectory in programfilesDirectories)
31:          {
32:                  var fullPath =
GetFileNameOfLocalDbManagementTool(programfilesDirectory);
33:
34:                      if (File.Exists(fullPath))
35:                      {
36:                            return fullPath;
37:                      }
38:          }
39:          throw new FileNotFoundException("Could not locate LocalDB admin
utility. Verify that LocalDB is installed."); 40:         }
41:
42:         private static string GetFileNameOfLocalDbManagementTool(string
programFilesDir)
43:         {
44:             return Path.Combine(programFilesDir, @"Microsoft SQL
Server\120\Tools\Binn\SqlLocalDB.exe");
45:         }
```

I Valg.EVA.Skanning.JobManagement.Startup.BuypassBootstrapper er det ikke bruk av strengkonstanter, men .NET-metoder for å finne «Program Files»-kataloger:

```
75:         private static string AttemptGetPathToBuypass()
76:         {
77:             var pathToBuypass =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ProgramFiles),
buypassAccessIidPath);
78:
79:             if (!File.Exists(pathToBuypass))
80:             {
82:                pathToBuypass =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ProgramFilesX86)
, buypassAccessIidPath);
83:             }
84:
85:             return pathToBuypass;
86:         }
```

Enironment.SpecialFolder gir også den lokale varianten av «Program Files»(x86) og her klarer koden seg med to forsøk på å finne katalogen, mens den varianten som bruker tekststrenger har 3. Burde det ikke i så fall være 4, for å ta høyde for norsk versjon av (x86)-katalogen?

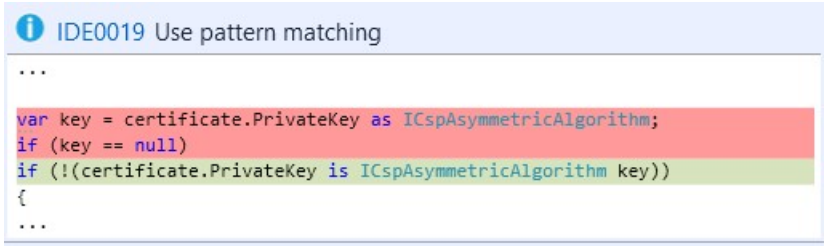
Når det gjelder feilhåndteringen har GetCorrectPathToTool() tatt høyde for at filen ikke kunne finnes. I AttemptGetPathToBuypass() brukes resultatet fra ProgramFilesX86 uten å sjekke om fila ligger på det stedet. Det vil nok være slik at bruk av .NET-metoder for å søke opp ressurser er mer robust enn å bruke strengkonstanter, men det er allikevel god praksis å sjekke at verdier som brukes faktisk inneholder det som er forventet.

*Anbefaling 5 Bruk .NET-funksjoner for å finne resurser der det er mulig*

*Anbefaling 6 Lag felles kodestandard og koderegler*

Moderne IDE-er kan hjelpe til med å få ensartet kode i prosjektet, og de hjelper til med å få koden til å følge nyere funksjonalitet i koden. Her har vi et par eksempler fra Visual Studio, hvor den forslår å bytte ut koden i rødt med det som er i grønt. Det vil (i disse eksemplene) både redusere kodemengden, samt gjøre det enklere å forstå.

I det første eksempelet vil «is» både gjøre variabeldeklarasjon, type- og null-sjekk:




I neste eksempel vil bruk av «auto property», som både gir variabeldeklarasjon og en mer kompakt definisjon av get() enn det som er i koden i dag:


Andre steder i koden brukes «auto property», og det er dermed to ulike stiler på koden for samme funksjonalitet.

*Anbefaling 7 Bruk Visual Studio til å hjelpe til med å følge kodestandard og for å redusere unødvendig verbos kode*

Det er ikke kun C#-kode og –bibliotek som må holdes oppdatert, men også definisjonsfiler for tilhørende funksjonalitet, inkl. EML-definisjoner og eventuelle definisjoner for annen XML-basert funksjonalitet.

Det ser ut til å være bruk av gammel versjon av XSD-fila for XML Signature i scanning\Eva\Scanning\SetupTool\ElectionConfigImport\Schema\external\xmldsig-core-schema.xsd

da den mangler definisjon av bl.a. X509CRL i listen over X509DataType-er. Vi ser ikke nødvendigvis noe behov for nettopp X509CRL i EVA Skanning, men vi ser heller ingen grunn til å bruke en eldre versjon av XSD-fila. Siste versjon er 1.2: https://www.w3.org/TR/xmldsig-core/xmldsig-coreschema.xsd

# 4 Om applikasjoner i «ukjent» infrastrukturmiljø

EVA Skanning skiller seg ut fra de to andre EVA-delene vi har analysert ved at det er en Windowsapplikasjon som kjører på PC-er i opptellingslokaler. Det er noen anbefalinger til hvordan disse PC-ene skal settes opp, samt at PC-er som gjenbrukes for ulike opptellinger nyinstalleres mellom hver telling. Det er dog ingen tekniske mekanismer som må følges.

Valgdirektoratet har ikke myndighet til å bestemme kjøringsmiljøet til EVA Skanning ute i fylkene/kommunene og selv om Valgdirektoratet skulle komme med tekniske krav vil det være mulig å omgå dette da dette er utstyr som er installert og kontrollert lokalt.

Vi har noen alternativer som vi ser, og som ikke nødvendigvis er nye for Valgdirektoratet, men hvor vi prøver å framheve trusler og muligheter.

I denne analysen diskuterer vi svært lite det lokale nettverket som EVA Skanning kjører i. Det er fordi nettverket vil være utenfor kontroll av Valgdirektoratet også om en får kontroll over PC-ene som kjører i nettet. Vi anser nettverket som utrygt (også i analysen av EVA Skanning generelt), f.eks. ved kommunikasjon mellom skanner-PC og database der hvor det kjører mer enn én skanner og hvor det brukes felles database. I den generelle analysen anser vi dog at PC-ene har en viss basissikring og vi har kun vektlagt lokale angrep på PC-en når det er enkelt å legge på beskyttelsesmekanisme, eller hvor eksisterende bruk av en slik beskyttelsesmekanisme er kombinert med noe som reduserer beskyttelseseffekten. Vi er mao. mer bekymret for databaseaksess i nettverket enn på lokal PC, og vi er mer bekymret for feil bruk av sikkerhetsmekanismer enn at EVA Skanning selv skal inneholde komplett beskyttelse mot lokale angrep, inkl. mottiltak mot ondartede program som kjører på samme PC. Når det er sagt er det selvsagt mulig å sette krav til overvåkning av trafikk i nettet, f.eks. ved å sette ut dedikert overvåkningsutstyr, med rapportering tilbake til sentral infrastruktur. Det vil være et naturlig kompliment til noen av de følgende alternativene for sikring av PC-er.

*Merk*: det er ikke kjent at EVA Skanning (eller noen av de andre komponentene) har vært utsatt for manipulasjon («hacking»). Vår jobb er dog å påpeke hvor det kan skje.

## 4.1 Sentral skannerinfrastruktur

Valgdirektoratet har selv vurdert å endre arkitektur til en mer sentralisert løsning, hvor verifisering og påfølgende steg kan gjøres på sentral infrastruktur. Det vil fortsatt være nødvendig å ha kjørende en komponent lokalt og som gjør selve skanningen. Vi har under sett for oss at filen som inneholder bildet av den skannede valgseddelen sendes til sentral infrastruktur.

Fordeler:

- Enklere konfigurasjon av lokal PC da det er færre komponenter som skal kjøres og konfigureres
- Manntallsinformasjon holdes i sentral infrastruktur
- Verifikasjon og opptelling skjer i sentral infrastruktur
- Bedre loggmuligheter, for både feilsøking og for å avdekke eventuelle hendelser  Mulig å gjøre tuning av OCR-skanningen i løpet av valget Ulemper/Trusler:
- Manglende kontroll på lokale PC-er som kjører selve skanningen (som i dag)

- Økt krav på nettverksforbindelse til sentral infrastruktur, både båndbredde og tilgjengelighet
- Økte ressursbehov i sentral infrastruktur
- Behov for sikkerhets- / virusskanning av innsendte skannefiler

Det er mulig med en mellomting, hvor OCR-delen fortsatt gjøres på lokal PC, men det vil gjøre det enklere å manipulere resultatet da det er mer informasjon som behandles lokalt før det sendes videre, og det vil derfor være enklere å angripe en slik løsning da det er både tekst- og billeddata som kan manipuleres, samt at det er noe mer tid mellom billedfilen blir skannet inn til dataene blir sendt enn om billedfilen blir sendt direkte. Et slikt angrep krever at angriper har fått kontroll over skann-PC-en.

## 4.2 Anse PC-ene som tilsvarende mobiler, og lag tekniske regler for minimumssikring

MDM (Mobile Device Management) og EMM (Enterprise Mobile Management) er et konsept for sikring av mobile enheter hvor det lages sentrale regler for hva som en mobil må ha av minimumskrav for å kunne kjøre utvalgte applikasjoner eller nå visse tjenester. MDM brukes gjerne i kombinasjon med BYOD (Bring Your Own Device), hvor eieren av mobilen får installert en MDM-klienten på sin mobil, og hvor det er denne som tilbyr ønsket funksjonalitet (tradisjonelt epost, nettleser mot intern bedriftsinformasjon, o.l.), og som sjekker at kravene for å kjøre er oppfylt.

Slik EVA Skanning-applikasjonene kjører i dag kan vi anse de lokale PC-ene som BYOD da det er lokale administratorer som styrer PC-en og ikke Valgdirektoratet eller departementet.

EVA Skanning er en Windows-applikasjon. Fra Windows 8.1 er støttet av Microsofts egen MDM/EMM-løsning, Intune. Intune bruker Azure for å kommunisere EMM-konfigurasjonen mellom sentral løsning og mobil/PC.

Fordeler:

- Det er mulig å bruke tekniske mekanismer for å sjekke at en PC som skal kjøre skanning har visse minimumskrav
- Kravene kan endres å rulles ut i løpet av valgprosessen Ulemper:
- Det er fortsatt lokalt installerte PC-er som vil kunne være manipulert og som
- Intune kan sette krav, men er ingen fullverdig konfigurasjonsløsning
- Intune er ingen overvåkningsløsning mot ondartet kode
- Bruk av Azure er en ny komponent som ikke er i bruk i dag

Referanse: https://docs.microsoft.com/en-us/intune/introduction-intune

## 4.3 Bruk tredjepartsverktøy for mer komplett konfigurasjonsstyring og eventuelt overvåkning

Det finnes ulike tredjepartsløsningen som tilbyr konfigurasjonsstyring av Windows-enheter, såkalt «endpoint protection». Foruten krav til hva som kjøres, og ikke kjøres, på en PC, er det også mulighet for utrulling av programvare (noe også MDM tilbyr), patch-management, kontroll av eksternt utstyr som tilkobles, m.m. Endpoint protection-løsninger gir bedre kontroll enn MDM-løsninger tilbyr.

«Endpoint detection»-løsninger er tenkt å tilby mer avanserte overvåkningsfunksjonalitet enn tradisjonell anti-virus, hvor fokus har vært på signaturbasert gjenkjenning. Med avansert overvåkning tenker vi her på å overvåke om prosesserer utfører potensielt skadelig funksjonalitet og rapportere og/eller blokkere slik oppførsel.

De to vinklingene over kan kombineres.

Fordeler:

- Gir bedre kontroll over hva som er tillatt og ikke tillatt å kjøre enn MDM/EMM  Gir mulighet for å avdekke forsøk på målrettede angrep mot EVA Skanning Ulemper:
- Lisenser og prismodell er dårlig egnet for valggjennomføring
- Lokale administratorer har fortsatt mulighet til å stoppe, reinstallere og endre PC-er som brukes for EVA Skanning
- Krever nettverksaksess for utrulling og for rask rapportering

## 4.4 Sende ut PC-er installert og konfigurert sentralt

For å få full kontroll over kjøringsmiljøet til EVA Skanning anser vi kun at lokalt installerte og konfigurerte PC-er, som distribueres til valg-/telle-/ administrasjonslokaler er eneste løsning.

Fordeler:

- Forenkler skannerkonfigurasjon i kretser, kommuner og fylker
- Full kontroll over hva som er installert på hver skann-PC, med ferdig installasjon av EVA Skanning
- Kan fjerne Administrator-tilgang på lokalt nivå
- Kan hente ut feillogger når PC-er returneres etter valg

Ulemper:

- Økt sentral arbeidsbyrde før valg
- Økte kostnader for sentraliserte PC-er
- PC-er vil samle støv i perioder, og vil kunne bli foreldet raskt i forhold til hvor lite brukt de blir (men forhåpentlig ikke i Tress-90-proporsjoner)  Beskytter ikke mot fysisk manipulasjon av PC-ene

# 5 Kryptorelaterte funn

Dette kapittelet inneholder kryptofunn som er spesifikke for EVA Skanning

## 5.1 Valg.EVA.Skanning.Signing.SignatureCryptographicServiceProvider

I SignData() brukes PKCS #1 v1.5:

```
77: return rsa.SignData(data, HashAlgorithmName.SHA256,
RSASignaturePadding.Pkcs1);
```

PKCS #1 v1.5-signaturer ble standardisert i RFC 2437 i 1998, PKCS-serien var opprinnelig publisert av RSA Security.

PKCS #1 v1.5 har hatt en del sikkerhetsproblemer, særlig knyttet til kryptering, men også signering. For signering er problemene som oftest implementasjons- eller bruks-spesifikke, og ikke generelle. På grunn av disse svakhetene ble OAEP (for kryptering) innført i PKCS #1 v2, og PSS (for signering) i PKCS #1 v2.1. Disse er mer robuste design som har vært utsatt for grundig teoretisk analyse, dermed med lavere risiko for at noe går galt. Nyeste versjon av PKCS #1 er versjon 2.2 fra 2016 (RFC 8017).

Fra RFC8017 er anbefalingen rundt signering som følger:

> *Two signature schemes with appendix are specified in this document:*
>
> *RSASSA-PSS and RSASSA-PKCS1-v1_5. Although no attacks are known against RSASSAPKCS1-v1_5, in the interest of increased robustness, RSASSA-PSS is REQUIRED in new applications. RSASSA-PKCS1-v1_5 is included only for compatibility with existing applications.*

Ettersom mekanismene fra versjon 1.5 kom på et veldig opportunt tidspunkt under .com-boblen, har de imidlertid blitt tatt i bruk i svært mange andre standarder, og de tar dermed lang tid å bli kvitt. Mange løsninger støtter ennå ikke PSS, og da er man avhengig av PKCS #1 v1.5 av kompabilitets-årsaker.

Dette kan i hovedsak anses som en «hazard» - det er litt frynsete i kantene, og i kombinasjon med uflaks eller med andre sårbarheter kan det bidra til større konsekvenser av uhell – snarere enn en «vulnerability» som kan utnyttes direkte i et angrep mot valgsystemene.

Vi har fått vite at bruk av v1.5 i EVA Skanning er pga en avhengighet til BuyPass, som ikke støtter v2. Vi anbefaler å ta kontakt med BuyPass og be om støtte for v2/PSS. *Anbefaling 8 Be leverandør (BuyPass) om å støtte PSS (PKCS1 v2)*

# 6 Verktøybaserte funn: komponenter med sårbarheter

Dette kapittelet inneholder funn fra OWASP Dependency Check-verktøyet. Se vedlagte råapporter for å få mer informasjon om hvilke versjoner som ikke er sårbare, hvilke CVE (Common Vulnerability Enumeration)-sårbarheter det dreier seg om, m.m.

Følgende .NET-bibliotek er i bruk i koden og har kjente sårbarheter:

- opencv_ffmpeg341.dll OpenCV versjon 3.4.1
- opencv_world310.dll OpenCV versjon 3.1.0

*Merk*: Den vedlagte råapporten har mange ekstra funn, men det ser ut til at disse er falske positive. Det kan være at innebygd støtte i NuGet for visning av nyere versjoner vil være et bedre alternativ for .NET-avhengigheter, selv om den ikke har kobling mot NISTs CVE-database, slik som OWASPverktøyet.

# 7 Verktøybaserte funn: kildekodefunn

Dette kapittelet inneholder detaljert informasjon om hvert enkelt funn, slik Fortify SCA rapporterer det.

Vi har prøvd å kvalitetssikre funnene ved å fjerne de som er falske positive. Det er dog ikke alltid lett å komme utenfra å forstå koden og det vil nok være funn som vi har tatt med, men som utviklerne vil kategorisere som falske positive. De færreste funnene er direkte utnyttbare, men vi mener at ved å følge anbefalingene vil det være mindre fare for sikkerhetsrelaterte hendelser, og vi bruker her «sikkerhet i dybden» som prinsipp. Funnene vi anser som reelle har vi kategorisert iht Fortifys 4 innebygde analysekategorier:

- Reliability Issue – Dette er svakheter som kan gjøre at applikasjonen ikke oppfører seg som den skal. Det er ofte relatert til fysiske begrensninger og gjelder gjern filaksess, nettverkskommunikasjon, minneproblemer, o.l.
- Bad Practice – Denne kategorien inneholder både sikkerhetsrelaterte sårbarheter og kvalitetsrelaterte.
- Suspicious – Her har vi funn som vi enten er usikre på om er utnyttbare, eller som vil kunne være én av flere faktorer som skal til for at en sårbarhet oppstår, mtp sikkerhet i dybden.

- Exploitable – Dette er funn som er direkte utnyttbare. Det kan være manglende validering og outputkoding (Injection-/Cross-Site Scripting-angrep), manglende transportsikring (klartekstkommunikasjon i nettverk), eller andre angrep som vil kunne skje uten at andre sikkerhetsmekanismer brytes. Det vil dog være ulikt hvor alvorlige disse er, f.eks. om det krever aksess til lokalt nettverk eller om det kan gjøres direkte fra Internett.

Et funn kan høre hjemme i flere kategorier, og det er særlig «Suspicious» hvor vi er usikre på om sårbarheten er reell eller ikke.

Funnene er sortert etter alvorlighetsgrad slik at de mest kritiske funnene kommer først.

## **[Medium]** Code Correctness / Missing [Serializable] Attribute

*Code Quality*

**Class ID:** 72610F51-FC46-4F0D-800D-B4CDEEEE9AFD

**Analysis:** Reliability Issue

### Abstract

Classes that implement the `ISerializable` interface but do not declare the `[Serializable]` attribute will not be serialized.

### Explanation

The .NET runtime will permit the serialization of any object that declares the `[Serializable]` attribute. If the class can be serialized using the default serialization methods defined by the .NET framework, this is both necessary and sufficient for the object to be correctly serialized. If the class requires custom serialization methods, it must also implement the `ISerializable` interface. However, the class must still declare the `[Serializable]` attribute.

**Example 1:** The `CustomStorage` class implements the `ISerializable` interface. However, because it fails to declare the `[Serializable]` attribute, it will not be serialized.

```
public class CustomStorage: ISerializable { ...
}
```

### Recommendation

Ensure that any class that implements custom serialization methods also implements both the `ISerializable` interface and declare the `[Serializable]` attribute. Classes that fail to declare the `[Serializable]` attribute will not be serialized.

**Example 2:** This example shows the code from Example 1 rewritten to include the proper declaration of the `[Serializable]` attribute required for the class to be properly serialized.

```
[Serializable] public class CustomStorage:
ISerializable { ...
}
```

## 7.1 DatabaseCompatibilityException.cs

**Issue** 60F8663B14E8C699F117DD6C67021C5E:

**Tag** Reliability Issue

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SharedProjects/Contracts/DatabaseCompatibilityException.cs 5-16

**Fact** Name: Valg.EVA.Skanning.Contracts.DatabaseCompatibilityException

```
  3|namespace Valg.EVA.Skanning.Contracts
  4|{
> 5|    public class DatabaseCompatibilityException : Exception > 6|    {
> 7|        public DatabaseCompatibilityException(string currentDatabaseVersion, string expectedDatabaseVersion)
> 8|        {
> 9|            CurrentDatabaseVersion = currentDatabaseVersion;
>10|            ExpectedDatabaseVersion = expectedDatabaseVersion;
>11|        }
>12|
>13|        public string CurrentDatabaseVersion { get; }
>14|        public string ExpectedDatabaseVersion { get; }
>15|
>16|        public override string Message =>  $"An exception of type { GetType() } was thrown. Current database
version: {CurrentDatabaseVersion}, Expected database version: {ExpectedDatabaseVersion}";
```

## **[Medium]** Missing XML Validation

*Input Validation and Representation*

**Class ID:** 341D18D4-39D9-41B6-AB33-954AE0CAD116

**Analysis:** Suspicious

### Abstract

Failure to enable validation when parsing XML gives an attacker the opportunity to supply malicious input.

### Explanation

Most successful attacks begin with a violation of the programmer's assumptions. By accepting an XML document without validating it against a DTD or XML schema, the programmer leaves a door open for attackers to provide unexpected, unreasonable, or malicious input. It is not possible for an XML parser to validate all aspects of a document's content; a parser cannot understand the complete semantics of the data. However, a parser can do a complete and thorough job of checking the document's structure and therefore guarantee to the code that processes the document that the content is well-formed.

### Recommendation

Always enable validation when you parse XML. If enabling validation causes problems because the rules for defining a well-formed document are Byzantine or altogether unknown, chances are good that there are security errors nearby.

**Example:** The following code demonstrates how to enable validation when using `XmlReader`.

```
XmlReaderSettings settings = new XmlReaderSettings(); settings.Schemas.Add(schema);
settings.ValidationType = ValidationType.Schema; StringReader sr = new
StringReader(xmlDoc);
XmlReader reader = XmlReader.Create(sr, settings);
```

## 7.2 Extensions.cs

**Issue** 3E16BECD582088F6A50D08ECF6D98DC2:

**Tag** Suspicious

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SetupTool/Extensions.cs 15 **Action** InCall Create()
**Rule id** 341D18D4-39D9-41B6-AB33-954AE0CAD116

```
 12|public static XmlReader WrapInXmlReader(this string stringToWrap)
13|{
 14|    var reader = new StringReader(stringToWrap);
>15|    return XmlReader.Create(reader);
 16|}
 17|
 18|public static dynamic ToDynamic(this object value, params string[] excludeProperties)
```

## **[Medium]** Password Management / Password in Configuration File

*Environment*

**Class ID:** 639058E8-DD47-47BE-BE56-E78399BB9683

**Analysis:** Bad Practice

### Abstract

Storing a plaintext password in a configuration file could result in a system compromise.

### Explanation

Storing a plaintext password in a configuration file allows anyone who can read the file access to the password-protected resource. Developers sometimes believe that they cannot defend the application from someone who has access to the configuration, but this attitude makes an attacker's job easier. Good password management guidelines require that a password never be stored in plaintext.

### Recommendation

A password should never be stored in plaintext. Instead, the password should be entered by an administrator when the system starts. If that approach is impractical, a less secure but often adequate solution is to obfuscate the password and scatter the de-obfuscation material around the system so that an attacker has to obtain and correctly combine multiple system resources to decipher the password.

Microsoft(R) provides a tool that can be used in conjunction with the Windows Data Protection application programming interface (DPAPI) to protect sensitive application entries in configuration files [1].

# 7.3 App.config

**Issue** EFB7EC00FC6C22F174EFCEC315DC7A5A:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/JobManagement/Startup/App.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

**Issue** EFB7EC00FC6C22F174EFCEC315DC7A5B:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Presentation/App.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
```

```
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

**Issue** EFB7EC00FC6C22F174EFCEC315DC7A5C:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Scan.Startup/App.config 4

```
 1|<?xml version="1.0" encoding="utf-8"?>
 2|<configuration>
 3|  <connectionStrings>
>4|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 5|  </connectionStrings>
 6|  <appSettings>
 7|    <add key="DatabaseVersion" value="3.1.206" />
```

**Issue** EFB7EC00FC6C22F174EFCEC315DC7A5D:

**Tag** Bad Practice

**Sink :**

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

# 7.4 Valg.EVA.Skanning.JobManagement.Startup.exe.config

**Issue** D8D331E364A37C88237A31596CEDD717:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/JobManagement/Startup/bin/Release/Valg.EVA.Skanning.JobManagement.Startup.exe.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

# 7.5 Valg.EVA.Skanning.Scan.Presentation.dll.config

**Issue** FFDEAE3B0DDF411388AB5CE95F82923C:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/FormManagement/FormDefinitionExport/bin/Release/Valg.EVA.Skanning.Scan.Presentation.dll.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

**Issue** FFDEAE3B0DDF411388AB5CE95F82923D:

### **Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/ReadSoftFormsAppImpostor/bin/Release/Valg.EVA.Skanning.Scan.Presentation.dll.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

## **Issue** FFDEAE3B0DDF411388AB5CE95F82923E:

### **Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Api/bin/Release/Valg.EVA.Skanning.Scan.Presentation.dll.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

## **Issue** FFDEAE3B0DDF411388AB5CE95F82923F:

### **Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Presentation/bin/Release/Valg.EVA.Skanning.Scan.Presentation.dll.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

## **Issue** FFDEAE3B0DDF411388AB5CE95F829240:

### **Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Scan.Startup/bin/Release/Valg.EVA.Skanning.Scan.Presentation.dll.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

# 7.6 Valg.EVA.Skanning.Scan.Startup.exe.config

## **Issue** 83633AD1091E08F6BB7AEEC62C10B88E:

### **Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Scan.Startup/bin/Release/Valg.EVA.Skanning.Scan.Startup.exe.config 4

```
 1|<?xml version="1.0" encoding="utf-8"?>
 2|<configuration>
 3|  <connectionStrings>
>4|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 5|  </connectionStrings>
 6|  <appSettings>
```

```
 7|    <add key="DatabaseVersion" value="3.1.206" />
```

## 7.7 Valg.EVA.Skanning.Verify.Startup.exe.config

**Issue** 2D6B00FEC3FFC010729D2F5D85A05A35:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Verify/Presentation/Startup/bin/Release/Valg.EVA.Skanning.Verify.Startup.exe.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

## **[Medium]** Weak Cryptographic Hash

*Security Features*

**Class ID:** 19032DF5-868A-43FC-B598-E0982E3FF3CF

**Analysis:** Falsk Positiv: Som nevnt i 3.1 er ikke CryptoOids lenger i bruk da ny kode bruker alternativ funksjonalitet og hele klassen kan fjernes. Gammel kode brukte forsåvidt heller ikke SHA1, så denne definisjonen kunne vært fjernet uansett.  Vi har dog tatt med funnet for å visualisere diskusjonen i avsnitt 3.1.

### Abstract

Weak cryptographic hashes cannot guarantee data integrity and should not be used in security-critical contexts.

### Explanation

MD2, MD4, MD5, RIPEMD-160, and SHA-1 are popular cryptographic hash algorithms often used to verify the integrity of messages and other data. However, as recent cryptanalysis research has revealed fundamental weaknesses in these algorithms, they should no longer be used within security-critical contexts.

Effective techniques for breaking MD and RIPEMD hashes are widely available, so those algorithms should not be relied upon for security. In the case of SHA-1, current techniques still require a significant amount of computational power and are more difficult to implement. However, attackers have found the Achilles' heel for the algorithm, and techniques for breaking it will likely lead to the discovery of even faster attacks.

### Recommendation

Discontinue the use of MD2, MD4, MD5, RIPEMD-160, and SHA-1 for data-verification in securitycritical contexts. Currently, SHA-224, SHA-256, SHA-384, SHA-512, and SHA-3 are good alternatives. However, these variants of the Secure Hash Algorithm have not been scrutinized as closely as SHA-1, so be mindful of future research that might impact the security of these algorithms.

## 7.8 CryptoOids.cs

**Issue** 6207533B2D2725E048FB217EA85E648D:

**Tag** Suspicious

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SharedProjects/Signing/CryptoOids.cs 83 **Action** InCall Oid()
**Rule id** 19032DF5-868A-43FC-B598-E0982E3FF3CF

```
>83|public static readonly Oid Sha1 = new Oid("1.3.14.3.2.26");
```

## **[Low]** ASP.NET Bad Practices / Leftover Debug Code

*Encapsulation*

**Class ID:** D17237D4-4682-4F5E-82E9-68C32AE6351C

**Analysis:** Bad Practice

### Abstract

Debug code can create unintended entry points in a deployed web application.

### Explanation

A common development practice is to add "back door" code specifically designed for debugging or testing purposes that is not intended to be shipped or deployed with the application. When this sort of debug code is accidentally left in the application, the application is open to unintended modes of interaction. These back door entry points create security risks because they are not considered during design or testing and fall outside of the expected operating conditions of the application.

The most common example of forgotten debug code is a `Main()` method appearing in a web application. Although this is an acceptable practice during product development, classes that are part of a production ASP.NET application should not define a `Main()`.

### Recommendation

Remove debug code before deploying a production version of an application. Regardless of whether a direct security threat can be articulated, it is unlikely that there is a legitimate reason for such code to remain in the application after the early stages of development.

# 7.9 Program.cs

**Issue** A6A4DF024B4AFB6E1835B28101C4EBB5:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SetupToolConsole/Program.cs 11-28
**Fact** Name: Valg.EVA.Skanning.SetupToolConsole.Program.Main

```
  8|{
  9|    class Program
 10|    {
>11|        static void Main(string[] args)
>12|        {
```

## **[Low]** Dead Code / Unused Field

*Code Quality*

**Class ID:** 709CA699-BD82-4786-B8F5-6BA421A1AB98

**Analysis:** Bad Practice

### Abstract

This field is never used directly or indirectly by a public method.

### Explanation

This field is never accessed, except perhaps by dead code. Dead code is defined as code that is never directly or indirectly executed by a public method. It is likely that the field is simply vestigial, but it is also possible that the unused field points out a bug.

**Example 1:** The field named `glue` is not used in the following class. The author of the class has accidentally put quotes around the field name, transforming it into a string constant.

```
public class Dead {

  string glue;

  public string GetGlue() {     return "glue";
  }

}
```

**Example 2:** The field named `glue` is used in the following class, but only from a method that is never called by a public method.

```
public class Dead {

  string glue;

  private string GetGlue() {     return glue;
  }

}
```

### Recommendation

In general, you should repair or remove dead code. To repair dead code, execute the dead code directly or indirectly through a public method. Dead code causes additional complexity and maintenance burden without contributing to the functionality of the program.

## 7.10 Certificates.cs

**Issue** 8C04C02B8A13C851F7961AF2C2BB9076:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SharedProjects/Signing/Certificates.cs 13 **Fact**
Name: Valg.EVA.Skanning.Signing.Certificates.RevocationMode

```
 10|{
 11|    public class Certificates : ICertificates
 12|    {
>13|        private const X509RevocationMode RevocationMode = X509RevocationMode.NoCheck;
 14|        private static readonly Collection<Func<X509Certificate2Collection, X509Certificate2Collection>>
FilterFunctions =
 15|            new Collection<Func<X509Certificate2Collection, X509Certificate2Collection>>();
```

## **[Low]** Dead Code / Unused Method

*Code Quality*

**Class ID:** AC21F232-1D82-49B7-9AB1-46FE84CD6424

**Analysis:** Bad Practice

### Abstract

This method is not reachable from any method outside the class.

### Explanation

This method is never called or is only called from other dead code. Dead code is defined as code that is never directly or indirectly executed by a public method.

**Example 1:** In the following class, the method `DoWork()` can never be called.

```
public class Dead {   private
void DoWork() {
Console.Write("doing work");
  }
  public static void Main(string[] args) {
    Console.Write("running Dead");
  } }
```

**Example 2:** In the following class, two private methods call each other, but since neither one is ever invoked from anywhere else, they are both dead code.

```
public    class    DoubleDead    {
private void DoTweedledee() {
    DoTweedledumb();
  }
  private void DoTweedledumb() {
    DoTweedledee();
  }   public static void Main(string[]
args) {
    Console.Write("running DoubleDead");
  }
}
```

(In this case it is a good thing that the methods are dead: invoking either one would cause an infinite loop.)

### Recommendation

A dead method may indicate a bug in dispatch code.

**Example 3:** If method is flagged as dead named `GetWitch()` in a class that also contains the following dispatch method, it may be because of a copy-and-paste error. The 'w' case should return `GetWitch()` not `GetMummy()`.

```
public ScaryThing GetScaryThing(char st)
{   switch(st) {     case 'm':
      return      GetMummy();
case 'w':
      return
GetMummy();
default:       return
GetBlob();
  }
}
```

In general, you should repair or remove dead code. To repair dead code, execute the dead code directly or indirectly through a public method. Dead code causes additional complexity and maintenance burden without contributing to the functionality of the program.

## 7.11 SignatureCryptographicServiceProvider.cs

**Issue** D57602061555797CF37BFE9D21E503FA:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SharedProjects/Signing/SignatureCryptographicServiceProvider.cs 82-91
**Fact** Name: Valg.EVA.Skanning.Signing.SignatureCryptographicServiceProvider.WritePemSignature

```
>82|        private static byte[] WritePemSignature(byte[] encodedSignature)
>83|        {
>84|            using (var stream = new StringWriter())
>85|            {
>86|                var info =
Org.BouncyCastle.Asn1.Cms.ContentInfo.GetInstance(Asn1Object.FromByteArray(encodedSignature));
>87|                var writer = new PemWriter(stream);
>88|                writer.WriteObject(info);
>89|                return Encoding.UTF8.GetBytes(stream.ToString());
```

## [Low] Poor Error Handling / Overly Broad Catch

*Errors*

**Class ID:** 638B1E12-45C7-47EF-8345-37D99BD995D2

**Analysis:** Bad Practice

### Abstract

The catch block handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

### Explanation

Multiple catch blocks can get ugly and repetitive, but "condensing" catch blocks by catching a high-level class like `Exception` can obscure exceptions that deserve special treatment or that should not be caught at this point in the program. Catching an overly broad exception essentially defeats the purpose of .NET's typed exceptions, and can become particularly dangerous if the program grows and begins to throw new types of exceptions. The new exception types will not receive any attention.

**Example:** The following code excerpt handles three types of exceptions in an identical fashion.

```
try {
  DoExchange();
}
catch (IOException e) {
  logger.Error("DoExchange failed", e);
}
catch (FormatException e) {
  logger.Error("DoExchange failed", e);
}
catch (TimeoutException e) {
  logger.Error("DoExchange failed", e);   }
```

At first blush, it may seem preferable to deal with these exceptions in a single catch block, as follows:

```
try {
  DoExchange();
}
catch (Exception e) {
  logger.Error("DoExchange failed", e);   }
```

However, if `DoExchange()` is modified to throw a new type of exception that should be handled in some different kind of way, the broad catch block will prevent the compiler from pointing out the situation. Further, the new catch block will now also handle exceptions of types `ApplicationException` and `NullReferenceException`, which is not the programmer's intent.

### Recommendation

Do not catch broad exception classes like `Exception`, `<SystemException>`, or `<ApplicationException>` except at the very top level of the program or thread.

# 7.12 ElectionConfigPackageValidator.cs

**Issue** 7A716304F2EA84951744EC324BE1AC8F:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SetupTool/ElectionConfigImport/CommandHandler/Package/Validation/ElectionConfigPackageValidator.cs 93-97

```
 90|            packageValidationResult.Errors.AppendLine(schemaValidationResult.Errors);
 91|        }
92|    }
>93|    catch
>94|    {
>95|        packageValidationResult.IsValid = false;
>96|        packageValidationResult.Errors.AppendLine(SetupLanguageString.XmlSchemaValidationFailedFor);
>97|        packageValidationResult.Errors.AppendLine(documentName);
```

# 7.13 MainWindowViewModel.cs

**Issue** 3F5DECD00EBD42740089F3A7F746214F:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SetupTool/MainWindowViewModel.cs 471-475

```
 468|        messageBox.ShowWindowWhenNoParentWindow(SetupLanguageString.MessageBox_Title_Error,
SetupLanguageString.ErrorUnableToGetFolderStructureFromDb, "", MessageBoxType.Error, sqlEx);
 469|        EvaSkanningEventSource.Log.GetFolderStructureForSqlServerFailed(sqlEx.ToString());  470|    }
>471|    catch (Exception ex)
>472|    {
>473|        var messageBox = new MessageBox.MessageBox(new MessageBoxViewModel());
>474|        messageBox.ShowWindowWhenNoParentWindow(SetupLanguageString.MessageBox_Title_Error,
SetupLanguageString.UnexpectedError, "", MessageBoxType.Error, ex);
>475|        EvaSkanningEventSource.Log.SettingSqlServerFileLocationFailed(ex.ToString());
```

**Issue** 7F57C70D57F1429EC0738D5CF3ECA73C:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SetupTool/MainWindowViewModel.cs 162-164

```
 159|    await DropDatabaseCommand.Execute(databaseConfiguration.GetConnectionStringMaster(),
evaSkanningInitialCatalog);
 160|    AppendOutputMessage($"{SetupLanguageString.TheDatabaseIsDeleted}");
161|}
>162|catch (Exception ex)
>163|{
>164|    possibleException = ex;
 165|}
 166|finally
 167|{
```

**Issue** 930981F32D9F0ED31E6B6DF551D752FD:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SetupTool/MainWindowViewModel.cs 204-209

```
 202|    AppendOutputMessage($"{SetupLanguageString.InstallationOfTheDatabaseIsCompleted}");
 203|}
>204|catch (Exception exception)
>205|{
>206|    EvaSkanningEventSource.Log.UpgradingEvaSkanningDatabaseFailed(databaseServer, databaseName,
exception.ToString());
>207|
>208|    AppendOutputMessage(SetupLanguageString.UpgradeFailed);
>209|    AppendOutputMessage(exception.Message);
 210|}
 211|finally
 212|{
```

**Issue** F4FFF8A8F32CC560C6877E6D57095C97:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SetupTool/MainWindowViewModel.cs 364-367

```
>364|    catch (Exception e)
>365|    {
>366|        var messageBox = new MessageBox.MessageBox(new MessageBoxViewModel());
>367|        messageBox.ShowWindowWhenNoParentWindow(SetupLanguageString.MessageBox_Title_Error,
SetupLanguageString.ErrorConnectionStringNotValid, "", MessageBoxType.Error, e);
```

## 7.14 SignatureCryptographicServiceProvider.cs

**Issue** D2B397D781A07365E890DD446DED8A64:

**Tag** Bad Practice

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/SharedProjects/Signing/SignatureCryptographicServiceProvider.cs 34-37

```
 31|        result.SignedData = SignData(data, certificate);
 32|        result.IsSigningSuccessful = true;
 33|    }
>34|    catch (Exception ex)
>35|    {
>36|        result.IsSigningSuccessful = false;
>37|        EvaSkanningEventSource.Log.SigningOfDataFailed(ex.ToString());  38|    }
 39|}
 40|else
```

## [Low] Insecure Transport / Database

*Security Features*

**Class ID:** 6D0BF387-7611-4B8E-AF59-5480D56C81AD

**Analysis:** Exploitable

### Abstract

The application is configured to communicate with its database server in plaintext over unencrypted channels, making the communicated data vulnerable to interception via man-in-the-middle (MiTM) attacks.

### Explanation

The application communicates with its database server over unencrypted channels and may pose a significant security risk to the company and users of that application. In this case, an attacker can modify the user entered data or even execute arbitrary SQL commands against the database server.

**Example 1:** The following configuration causes the application to communicate with its database server over unencrypted channels:

```
<connectionStrings>
  <add name="Test" connectionString="Data Source=210.10.20.10,1433; Initial Catalog=myDataBase;User
ID=myUsername;Password=myPassword;" providerName="System.Data.SqlClient" />
</connectionStrings>
```

### Recommendation

Most database servers offer encrypted alternatives on different ports that use SSL/TLS to encrypt all the data being sent over the wire. Always use these alternatives when possible.

**Example 2:** The following configuration causes the application to communicate with its database server over encrypted channels:

```
<connectionStrings>
  <add name="Test" connectionString="Data Source=210.10.20.10,1433; Initial Catalog=myDataBase;User
ID=myUsername;Password=myPassword; Encrypt=yes;" providerName="System.Data.SqlClient" /> </connectionStrings>
```

# 7.15 App.config

**Issue** 0C503390E38C1BACC8C7EB01447BC2F3:

**Tag** Exploitable

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/JobManagement/Startup/App.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

**Issue** 0C503390E38C1BACC8C7EB01447BC2F4:

**Tag** Exploitable

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Presentation/App.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

**Issue** 0C503390E38C1BACC8C7EB01447BC2F5:

**Tag** Exploitable

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Scan.Startup/App.config 4

```
 1|<?xml version="1.0" encoding="utf-8"?>
 2|<configuration>
 3|  <connectionStrings>
>4|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 5|  </connectionStrings>
 6|  <appSettings>
 7|    <add key="DatabaseVersion" value="3.1.206" />
```

**Issue** 0C503390E38C1BACC8C7EB01447BC2F7:

**Tag** Exploitable

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Verify/Presentation/Startup/App.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

# 7.16 Valg.EVA.Skanning.JobManagement.Startup.exe.config

**Issue** B6F8B465802D63ED33D0468607B6BDA3:

**Tag** Exploitable

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/JobManagement/Startup/bin/Release/Valg.EVA.Skanning.JobManagement.Startup.exe.config 5

```
 2|<configuration>
 3|  <connectionStrings>
```

```
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

# 7.17 Valg.EVA.Skanning.Scan.Presentation.dll.config

## **Issue** B56FDDC4F6F839176C45F1174A57DECA:

## **Tag** Exploitable

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/FormManagement/FormDefinitionExport/bin/Release/Valg.EVA.Skanning.Scan.Presentation.dll.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

## **Issue** B56FDDC4F6F839176C45F1174A57DECB:

## **Tag** Exploitable

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/ReadSoftFormsAppImpostor/bin/Release/Valg.EVA.Skanning.Scan.Presentation.dll.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

## **Issue** B56FDDC4F6F839176C45F1174A57DECC:

## **Tag** Exploitable

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Api/bin/Release/Valg.EVA.Skanning.Scan.Presentation.dll.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

## **Issue** B56FDDC4F6F839176C45F1174A57DECD:

## **Tag** Exploitable

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Presentation/bin/Release/Valg.EVA.Skanning.Scan.Presentation.dll.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

## **Issue** B56FDDC4F6F839176C45F1174A57DECE:

## **Tag** Exploitable

**Sink :**
**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Scan.Startup/bin/Release/Valg.EVA.Skanning.Scan.Presentation.dll.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

# 7.18 Valg.EVA.Skanning.Scan.Startup.exe.config

**Issue** 7A4FCDD61E87EFB3649E1988EE001322:

**Tag** Exploitable

**Sink :**

**File** E:/EVA-Scanning/scanning/Eva/Scanning/Scan/Presentation/Scan.Startup/bin/Release/Valg.EVA.Skanning.Scan.Startup.exe.config 4

```
 1|<?xml version="1.0" encoding="utf-8"?>
 2|<configuration>
 3|  <connectionStrings>
>4|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 5|  </connectionStrings>
 6|  <appSettings>
 7|    <add key="DatabaseVersion" value="3.1.206" />
```

# 7.19 Valg.EVA.Skanning.Verify.Startup.exe.config

**Issue** FA9D91E69B64742C80B6002E5B460B7C:

**Tag** Exploitable

**Sink :**

**File** E:/EVA-Scanning/scanning/Eva/Scanning/Verify/Presentation/Startup/bin/Release/Valg.EVA.Skanning.Verify.Startup.exe.config 5

```
 2|<configuration>
 3|  <connectionStrings>
 4|    <clear />
>5|    <add name="ScanningDatabase" connectionString="Data Source=(local);Initial Catalog=EvaScanning;User
Id=EvaScanning;Password=ev@SCANN1ng;" />
 6|  </connectionStrings>
 7|  <appSettings>
 8|    <add key="DatabaseVersion" value="3.1.206" />
```

# Innledning

Valgdirektoratet arbeider kontinuerlig med å forbedre sine systemer, koderevisjoner er en del av denne forbedringsprosessen og gir viktige innspill for bedring av kodekvalitet og økt sikkerhet. Våren 2018 bestilte Valgdirektoratet koderevisjon for EVA applikasjonene EVA Admin, EVA Skanning og EVA Resultat fra en ekstern markedsaktør.

# Målsetning

Målsetningen for oppdraget var å få en uavhengig vurdering av kildekode og systemarkitektur for EVA-porteføljen der svakheter belyses langs to akser:

- *Sikkerhet* – Med sikkerhet menes feil eller mangler i applikasjonskode og/eller arkitektur som vil kunne ha konsekvenser for valgresultatet i den forstand at kode og/eller arkitektur åpner for/ikke tilstrekkelig utelukker 3. parts manipulasjon.
- *Kvalitet* – Med kvalitet menes feil eller mangler i kode og/eller arkitektur som er potensielle feilkilder og/eller som reduserer forståelse, testbarhet og vedlikeholbarhet. Typiske kandidater:
    - Kodekompleksitet
    - Kodeorganisering
    - Feil eller uhensiktsmessig bruk av biblioteker / rammeverk etc.

Vurderingene skulle brukes i forbedringsarbeidet av hele porteføljen for EVA

# Koderevisjonsprosess

Leverandør og Valgdirektoratet gjennomførte oppstartsmøte med innføring i applikasjoner og domene før leverandør gjorde sin revisjon og analyse. Ved oppdragets avslutning gikk leverandør gjennom funnrapporter med Valgdirektoratet.

Funnene fra leverandør ble så gjennomgått og vurdert av Valgdirektoratet, der Valgdirektoratet gjorde sin vurdering basert på fra alvorlighetsgrad angitt av leverandør. Utfallet av analysen var tredelt:

- Enkelte funn var falske positiver, dvs. at funnene ikke var reelle funn når kontekst for funnene ble vurdert, disse ble avvist
- De mest alvorlige funnene ble prioritert og rettet før produksjonssetting
- Mindre alvorlige funn ble prioritert ned og inngår i den generelle forbedringsprosessen for EVA produktene. Dette er funn som ikke er direkte sårbarheter, og som vurderes og prioriteres i videre utviklingsløp.

Valgdirektoratet er med denne prosessen trygg på at EVA-applikasjonene holder god kvalitet og er tilstrekkelig sikret for valget i 2019.

# Avviksrapport – EVA Skanning

## Anbefalinger – Kodekvalitet og arkitektur

### *Anbefaling 1 Skaff oversikt over alternativer til Unity-rammeverket*

Valgdirektorratet har foreløpig ikke vurdert andre alternativer, Unity-rammeverket er fremdeles aktivt vedlikeholdt. Vurderingen vil tas opp igjen knyttet til aktiviteter mot neste valg.

### *Anbefaling 2 Bruk mulighetene git gir for å modifisere én og én funksjonsendring*

Tatt til følge.

### *Anbefaling 3 Fjern alle referanser til Scytl*

Gjennomført.

### Anbefaling 4 Lag et system som gjør at det er lett å finne fram til tekstkonstanter som koden bruker

Forbedret.

### *Anbefaling 5 Bruk .NET-funksjoner for å finne resurser der det er mulig*

Skyldes ReadSoft forms løsningen som ikke lenger er i bruk.

### *Anbefaling 6 Lag felles kodestandard og koderegler*

Valgdirektoratet har en felles kodestandard, en del av koden er imidlertid arvet, forbedringer gjøres når funksjonsområder endres.

### *Anbefaling 7 Bruk Visual Studio til å hjelpe til med å følge kodestandard og for å redusere unødvendig verbos kode*

Valgdirektoratet har en felles kodestandard, en del av koden er imidlertid arvet, forbedringer gjøres når funksjonsområder endres.

### *Anbefalinger – kryptorelaterte funn*

### *Anbefaling 8 Be leverandør (BuyPass) om å støtte PSS (PKCS1 v2)*

Overføring av telling *signeres*, men *krypteres* ikke med Buypass smartkort. Det foreligger ikke planer fra Buypass om å endre krypertingsstandard for modulen vi bruker til *signering.*

# Verktøybaserte funn

## Komponenter med sårbarheter

Sårbarhetene beskrevet i rapporten er utbedret der nyere versjon finnes, eller der oppgradering til nyere versjon ikke hindres (se reelle funn under).

Sårbarhetene påpekt i OpenCv gjelder mating av ekstremt store bilder (1 GB eller mer), I EVA Skanning er det kun skanneren som leverer bilder og disse er av kjent størrelse. Valgdirektoratet vurdere et angrep med denne vinklingen som lite sannsynlig samtidig som det vil være lett å oppdage.

Valgdirektoratet har automatisert owasp i CI/CD byggpipeline som anbefalt av Mnemonic. Owasp-funn vurderes løpende i utviklingsprosessen som følger:

- Falske positiver / ikke direkte sårbarheter: Sårbarheter i biblioteker som brukes av biblioteker som brukes av EVA Skanning, men der underliggende biblioteks funksjonalitet ikke brukes
- Reelle funn:
    - Oppgraderes direkte dersom det finnes nyere versjon tilgjengelig og dersom konsekvensene av oppgradering er kontrollerbare
    - Planlegges dersom det finnes nyere versjon tilgjengelig, men der konsekvensene av oppgradering ikke er kontrollerbare
    - Legges på vent dersom nyere versjon ikke er tilgjengelig

# Verktøybaserte funn

Kildekodefunn
Avvikslogg – Utbedrede funn

| Avvik | Kommentar |
|---|---|
| Code Correctness / Missing [Serializable] Attribute<br>Analysis: Reliability Issue<br>Abstract Classes that implement the ISerializable interface but do not declare the [Serializable] attribute will not be serialized. | Lagt til serializable attributt |
| Password Management / Password in Configuration File<br>Analysis: Bad Practice<br>Abstract Storing a plaintext password in a configuration file could result in a system compromise | Passord krypteres med Windows Data Protection API |
| Dead Code / Unused Field<br>Analysis: Bad Practice<br>Abstract This field is never used directly or indirectly by a public method | Ryddet |
| Dead Code / Unused Method<br>Analysis: Bad Practice<br>Abstract This method is not reachable from any method outside the class | Ryddet |
| Poor Error Handling / Overly Broad Catch<br>Analysis: Bad Practice<br>Abstract The catch block handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program | Utbedret I den grad det er mulig. Tidligere Readsoft Forms føringer gjør seg fremdeles gjeldene til en viss grad. |
| Insecure Transport / Database<br>Analysis: Exploitable<br>Abstract The application is configured to communicate with its database server in plaintext | Se teknisk veileder I bruk av EVA Skanning for kommuner og fylkeskommuner kap. 3.1.8<br>Verktøy for kryptering av nettverk ved hjelp av selvsignert sertifikat følger installasjonspakkene. Sertifikat fra sertifikatutsteder anbefales. |

| | |
|---|---|
| over unencrypted channels, making the communicated data vulnerable to interception via man-in-the-middle (MiTM) attacks | |

## Avvikslogg – False positiver / ikke relevante funn.

| Avvik | Kommentar |
|---|---|
| Missing XML Validation<br>Analysis: Suspicious<br>Abstract Failure to enable validation when parsing XML gives an attacker the opportunity to supply malicious input | Funnet gjelder import av EML i forbindelse med installasjon / oppsett av EVA Skanning. EML er generert i EVA Admin – vurdert som ikke kritisk |
| Extensions.cs<br>Analysis: Bad Practice<br>Abstract Storing a plaintext password in a configuration file could result in a system compromise. | Funnet gjelder verktøy som brukes ved installasjon / oppsett – vurdert som ikke kritisk |
| Weak Cryptographic Hash<br>Analysis: Falsk Positiv: Som nevnt i 3.1 er ikke CryptoOids lenger i bruk da ny kode bruker alternativ funksjonalitet og hele klassen kan fjernes. Gammel kode brukte forsåvidt heller ikke SHA1, så denne definisjonen kunne vært fjernet uansett. Vi har dog tatt med funnet for å visualisere diskusjonen i avsnitt 3.1.<br>Abstract Weak cryptographic hashes cannot guarantee data integrity and should not be used in security-critical contexts. | Ingen aksjon |
| ASP.NET Bad Practices / Leftover Debug Code<br>Analysis: Bad Practice<br>Abstract Debug code can create unintended entry points in a deployed web application | Konsollprogram, ikke ASP.NET – vurdert som ikke relevant |