

# Självständigt arbete på grundnivå

*Independent degree project - first cycle*

Datateknik  
*Computer Engineering*

**Säkerhet**  
IT säkerhet på en myndighet

**Mikael Falck**



**Mittuniversitetet**

MID SWEDEN UNIVERSITY

Campus Härnösand Universitetsbacken 1, SE-871 88. Campus Sundsvall Holmgatan 10, SE-851 70 Sundsvall.

Campus Östersund Kunskapens väg 8, SE-831 25 Östersund.

Phone: +46 (0)771 97 50 00, Fax: +46 (0)771 97 50 01.

**MITTUNIVERSITETET**  
**Informations- och kommunikationssystem**

**Examinator:** Ulf Jennehag, [Ulf.jennehag@miun.se](mailto:Ulf.jennehag@miun.se)  
**Handledare:** Stefan Forsström, [Stefan.Forsstrom@miun.se](mailto:Stefan.Forsstrom@miun.se)  
**Författare:** Mikael Falck, [mifa1100@student.miun.se](mailto:mifa1100@student.miun.se)  
**Utbildningsprogram:** Civilingenjör i datateknik, 300 hp  
**Huvudområde:** Datateknik  
**Termin, år:** VT, 2015

## Sammanfattning

Detta projekt går ut på att hitta IT-säkerhetshot och hitta skyddsåtgärd för att förhindra en sådan attack. Syftet är att identifiera hoten och hitta en lösning för att förhindra sådana attacker mot myndigheten Bolagsverket. Slutleveransen ska vara en skyddsåtgärd mot ett specifikt hot som ska testas och verifieras praktiskt. Genom undersökningar i form av enkät och litteraturstudie samt intervjuer fysiskt på Bolagsverket, utreddes vilka hot som fanns. Hoten räknades sedan till risker via en riskanalys som också blev en prioritering. Via diskussion med handledare på Bolagsverket och med hänsyn till de prioriterade hoten så valdes en risk ut att arbeta, den risk som valdes att arbeta vidare med var okrypterad programkod. Det visade sig vara bra att den var okrypterad på grund av hur Java är utvecklat, skyddsåtgärd togs fram genom att göra koden svårförstådd med ett komplicerande -verktyg i kombination med att signera koden för att kunna kontrollera att den är oförändrad. Vidare använder de sig av en JBoss Enterprise Application Platform (EAP), vilket vidare undersöktes för att hitta skyddsåtgärd via JBoss EAP. Som har ett konfigurationsalternativ som kallas "secure-domains" för att hantera dess paketsäkerhet. Java kod signeringen och komplicerade-verktyget testades och verifierades att det fungerade, dock den svårlästa koden är inte mer än svårförstådd kod, utan det är signaturen som är intressant för att se till att koden är oförändrad. Tyvärr på grund av tidsbrist gick det inte att testa och verifiera de olika inställningarna som finns i "secure-domains" för JBoss.

**Nyckelord:** IT-säkerhet, verksamhetsanalys, riskanalys, gap-analys, Java, JBoss EAP.

## Abstract

The project is to find IT-security threats and to find protection against these to try and stop an attack in such nature as the threats. The Purpose is to identify the threats against Bolagsverket and a solution against one of the threats that been selected by prioritize the threats and a discussion with supervisors. The chosen threat was selected to be unencrypted source code on the production server. But because of the way Java was created it was not a good solution, as such the solution was found by obfuscating the code and use a code signer to check that the code haven't been manipulated by outside sources. Found that they are using a JBoss enterprise Application Platform (EAP) as application server for the Java code and tried to find some secure settings for the code and found "secure-domains" that is the main way to secure the container packages. Test and verifying the obfuscater and the code signing was done and successful but because of the time limit a test and verification of the "secure-domains"'s setting was not done.

**Keywords:** IT-security, information analysis, risk analysis, gap analysis, Java, JBoss EAP.

## Förord

Börjar med att tacka Martin Bylund och Mikael Larsson Österström från Bolagsverket, för att vara handledare åt författaren. De har också varit med och diskuterat projektet vid avstämningsmöten. Vill även tacka Stefan Forsström som handledare från Mittuniversitet i Sundsvall samt Stefan Ellström för hjälp med frågor om sekretess och ställt upp på en intervju. Även tack till Robert Strandberg för hjälp med riskanalysen för att bestämma konsekvens och sannolikhetsnivå för individuella hot. Till sist, stort tack till många andra på Bolagsverket, som ställt upp på enkätundersökningen och det trevliga mottagandet.

# Innehållsförteckning

<b>Sammanfattning</b> .....	<b>iii</b>
<b>Abstract</b> .....	<b>iv</b>
<b>Förord</b> .....	<b>v</b>
<b>Terminologi</b> .....	<b>viii</b>
<b>1 Inledning</b> .....	<b>1</b>
1.1 Bakgrund och problemmotivering.....	1
1.2 Övergripande syfte.....	1
1.3 Konkreta och verifierbara mål.....	1
1.4 Avgränsningar.....	2
1.5 Översikt.....	2
1.6 Författarens bidrag.....	2
<b>2 Teori</b> .....	<b>3</b>
2.1 Informationssäkerhet.....	3
2.2 Teknisk referensram.....	3
<b>3 Metod</b> .....	<b>5</b>
3.1 Verksamhetsanalys.....	5
3.2 Existerande IT-hot.....	6
3.3 Riskanalys.....	6
3.4 Gap-Analys.....	6
3.5 Framtagande av skyddsåtgärd.....	6
3.6 Test och verifikation av skyddsåtgärd.....	7
<b>4 Resultat</b> .....	<b>8</b>
4.1 Verksamhetsanalys.....	8
4.2 Existerande IT-hot.....	8
4.3 Riskanalys.....	10
4.4 Gap-analys.....	11
4.5 Framtagande av skyddsåtgärd.....	12
4.5.1 Skyddsåtgärd på jar filer.....	12
4.5.2 Skyddsåtgärder på JBoss EAP.....	14
4.6 Test och verifikation av skyddsåtgärd.....	15
<b>5 Diskussion</b> .....	<b>17</b>
5.1 Verksamhetsanalys.....	17
5.2 Existerande IT-hot.....	17
5.3 Riskanalys.....	18
5.4 Gap-analys.....	18
5.5 Framtagande av skyddsåtgärd.....	18
5.6 Test och verifikation av skyddsåtgärd.....	19
5.7 Förutsatta mål.....	19

---

5.8	Framtida forskning.....	20
5.9	Etiska aspekter.....	20
	<b>Källförteckning.....</b>	<b>21</b>
	<b>Bilaga A: verksamhetsanalys intervju frågor.....</b>	<b>23</b>
	<b>Bilaga B: enkät undersökning för IT-hot.....</b>	<b>24</b>
	<b>Bilaga C: exempel mall på risk dokument för ett individuellt hot.....</b>	<b>26</b>
	<b>Bilaga D: intervjufrågor för gap-analys.....</b>	<b>28</b>
	<b>Bilaga E: ursprung kod innan komplicerade verktyg används.....</b>	<b>29</b>
	<b>Bilaga F: Programkod efter komplicerade-verktyget har körts på den.....</b>	<b>30</b>
	<b>Bilaga G: alternativa listor för inställningar i ”secure-domain” för Jboss. 34</b>	
	Autentisering.....	34
	Autentisering-JASPI.....	35
	Tillstånd.....	35
	Kartläggning.....	36
	Granskning.....	36
	JSSE.....	37
	<b>Bilaga H: risker och konsekvensförklaring för vardera hot.....</b>	<b>39</b>
	Överbelastning (t,ex d-dos).....	39
	Virus.....	40
	Wifi.....	41
	Webb sidor.....	42
	Dator stöld.....	43
	Operativ systems.....	44
	Fysik installation av skadlig kod.....	45
	Registerbevis.....	46
	Säkras koden som används med kryptering.....	47
	Lönesystem.....	48
	Nätverks intrång.....	49
	Hot och säkerhetsproblem som eventuellt saknas.....	50

## Terminologi

DDoS:	”Distributed Denial of Service”.
EAP:	”Enterprise Application Platform”.
EE:	“Enterprise Edition”.
IBM:	“International Business Machines”.
ISP:	”Internet Service Provider”.
IT:	”Information Technology” informations teknik på svenska - Vanlig förkortning för dator relaterade produkter och enheter.
JAR:	”Java Archive”
JASPI:	”Java Authentication Service Provider”.
JSSE:	”Java Secure Socket Extension”.
JVM:	”Java Virtual Machine”.
MSB:	Myndigheten för Samhällsskydd och Beredskap.
SE:	”Standard Edition”.
XML:	“Extensible Markup Language”.

# 1 Inledning

Syftet med arbetet är att göra en initial grundkartläggning av de it- säkerhetshot som Bolagsverket behöver skydda sig mot. Ett utvalt hot analyseras och metoder för att verifiera skyddsåtgärder mot hotet utvecklas och verifieras genom praktisk test.

## 1.1 Bakgrund och problemmotivering

Bolagsverket är en myndighet som hanterar företag på en stor skala. Deras uppgifter är många. Några exempel på dessa är: registrering av företag, ta emot årsredovisningar, registrering av företagshypotek, fatta beslut om likvidation, tillhandahåller företagsfakta och mycket mer. En myndighet med denna storlek och vikt kommer alltid vara utsatt för hot utifrån.

Dessa hot mot Bolagsverket är något känsligt eftersom de påverkar alla företag i Sverige och konsekvenser av hoten kan vara mycket allvarliga för det svenska näringslivet och myndigheten.

## 1.2 Övergripande syfte

Syftet inom detta projekt är att utföra en initial grundkartläggning av it-säkerhetshot som Bolagsverket behöver skydda sig mot. Genom kartläggningen ska de hot som framkommit analyseras för att få en prioritering, utifrån prioriteringen på dessa hot ska ett väljas att arbeta vidare med. Det utvalda hotet ska analyseras för att hitta skyddsåtgärden som sedan ska verifieras med praktiska tester.

Ökad säkerhet på Bolagsverket är etiskt rätt för att företag ska känna en trygghet i att den information de lämnar ut är säker. Detta är viktigt för det svenska näringslivet på sådant sätt att företag vill ha kvar deras verksamheter i Sverige, vilket leder till att arbeten finns kvar i landet så befolkningen kan försörja sig.

## 1.3 Konkreta och verifierbara mål

1. Kartläggning av eventuella IT-hot. Detta innebär att det ska undersökas vilka hot som finns mot Bolagsverket. Det ska göras på en övergripande nivå. Varje hot ska få en sammanfattning som ska innehålla riskerna som hotets blev givet.
2. När kartläggningen är klar ska den diskuteras och prioriteras för att välja ut hot att arbeta vidare med.

3. Det valda hotet från mål två ska analyseras för att finna möjliga skyddsåtgärder, samtliga analyserade skyddsåtgärder ska dokumenteras och beskrivas.
4. De skyddsåtgärden som tagits fram från mål tre ska testas och verifieras praktiskt.

## 1.4 Avgränsningar

En avgränsning är att endast hot som finns mot IT kommer att utvärderas. För att arbetet inte ska bli för mycket för en person att genomföra, kommer inte alla hot behandlas, utan ett av de framtagna hoten kommer att väljas för att hitta skyddsåtgärd som ska förhindra detta hot.

## 1.5 Översikt

Kapitel 2 beskriver kunskaper som är viktiga för att förstå övrig text för läsaren. Kapitlet 3 beskriver tillvägagångssätt och metoder som används för det uppnådda resultatet. Kapitel 4 visar resultaten som uppnåtts och kapitel 5 är diskussion över resultatet och arbetet.

## 1.6 Författarens bidrag

Författaren har gjort allt på egen hand utom de nämnda sakerna nedan.

Martin Bylund och Mikael Larsson Österström har hjälpt till genom att välja hot samt varit med och avgjort om nästa steg är okej.

Stefan Ellström har ställt upp på intervju samt hjälpt till att kontrollera rapporten så inget sekretessbelagt skrivits ned.

Robert Strandberg har hjälpt till med att utföra riskanalysen.

## 2 Teori

Kapitel 2 ger grundläggande kunskaper som läsaren behöver för vidare läsning av rapporten. Underkapitel 2.1 förklarar informationssäkerhet på en översiktlig nivå och underkapitel 2.2 förklarar vilken teknik som använts.

### 2.1 Informationssäkerhet

Information är viktigt för vårt samhälle, idag så kan vi lagra och kommunicera vår information som aldrig förr. [1]

En stor del av informationen är viktig och i vissa fall livsviktig för oss. Exempel är patientjournaler eller styrsystem för kärnkraftverk. Då måste informationen skyddas enligt följande punkter: [1]

- Den ska vara lättillgänglig, finnas när den behövs (tillgänglighet). [1]
- Innehållet får inte vara förändras eller förstöras (riktighet). [1]
- Enbart personer med rätt behörighet kan komma åt informationen (konfidentialitet). [1]
- Ska vara möjligt att se när, eller hur, informationen visas/visats (spårbarhet). [1]

Dessa punkter och funderingar måste bli anpassade till den situation de ska tillämpas för. Då information är viktigt för dagens organisationer är det viktigt att hålla den säker så ingen obehörig kan förändra/förstör informationen. Genom att arbeta systematiskt med informationssäkerheten kan organisationen öka kvaliteten och förtroendet för verksamheten. Informationssäkerhet är administrativa rutiner med policys och riktlinjer samt tekniska skydd som exempelvis brandväggar och kryptering. [1]

### 2.2 Teknisk referensram

- Klassladdare är ett objekt som har hand om att ladda klasserna. Klassladdaren är en abstrakt klass, givet det binära namnet på en klass ska klassladdaren försöka hitta eller skapa data som definierar denna klass. [2]
- JD är en avkodare för .jar och .class filer skrivna i java. Detta innebär att personer kan få ut Java koden i klartext från den komprimerade byte koden. [3]

- Komplicerande-verktyg är ett verktyg som gör det svårare att förstå den skrivna koden, den byter namn på klasser från till exempel MySecretClass() till A(), samt byter namn på variabelnamn och metoder på liknade sätt. Det vill säga, programmet är inte komplicerat, utan programmet gör koden komplicerad och svår läst. [4]
- Proguard är ett komplicerande-verktyg (se punkt 5) som har används i testerna [5]. Detta är ett gratisverktyg som rekommenderas bland annat av IBM [6]. Proguard kan också minska storleken på koden för lagringsmedium samt minne, den procentuella mängden som minskas är dock beroende på ursprungskoden [7].
- Jarsigner är ett program som signerar Java kod med en publik- och privatnyckel vilket genereras av annan programvara. [8]
- Keytool är ett verktyg som hanterar och skapar privat- och publiknycklar. Detta är det program som Oracle använder i deras exempel för Jarsigner. Keytool är kort sagt en krypterad databas för nycklar som även kan generera nycklar på egen hand. [9]
- JBoss Enterprise Application Platform (EAP) är en kraftig java applikationsserver för Java EE 6. JBoss kommer med molnklar arkitektur och med kraftfulla hanteringsverktyg samt automatiseringar. JBoss EAP har Java EE 6 certifikat och har kraftfulla funktioner samtidigt är mycket flexibelt. JBoss EAP 6 är utvecklat för att minska underhåll, öka säkerhetsmöjligheter och är gjort för att vara resurs-snålt. Det ska vara enkelt att skicka meddelande (Java meddelanden) och ska vara snabbinstallerat för serverar och utvecklingsdatorer. [10]

## 3 Metod

De metoder som används härstammar från myndigheten för samhällsskydd och beredskap (MSB). MSB har en hemsida dedikerad för informationssäkerhet där de har lagt upp metodstöd för denna typ av undersökningar [11][12]. Dessa metoder har sedan blivit anpassade för detta projekt efter de resurser och tiden som har funnits till bifogande.

Metod ett och två kommer att användas för mål ett, metod tre kommer att användas för mål två, metod fyra och fem kommer att användas för mål tre och metod sex kommer att användas för mål fyra. De olika metoderna finns listat nedan:

1. Verksamhetsanalys.
2. Existerande IT-hot.
3. Riskanalys.
4. Gap-Analys.
5. Framtagande av skyddsåtgärd.
6. Test och verifikation av skyddsåtgärd.

### 3.1 Verksamhetsanalys

Verksamhetsanalysen utförs för att få en bild över vilken information som kan vara av intresse vid IT-attack mot Bolagsverket, ska även ge en inblick i dagsläget säkerhetsarbete. Då det är en myndighet som arbetet genomförts åt har organisationen redan säkerhetskrav enligt lag som måste följas.

Då det redan finns säkerhetskrav, har en intervju med säkerhetssamordnaren genomförts, vilket gav en bild om dagens säkerhetsarbete. Frågorna från intervjun finns i bilaga A.

Efter intervjun har en egen studie genomförts för att gå igenom blanketter som organisationer och kunder skickar in till Bolagsverket för se om det är någon information som Bolagsverket inte tänkt på som behöver skyddas. Denna information jämfördes sedan med vilken information som kan tas ut mot betalning/gratis för registrerade organisationer, för att se vilken information som inte är tänkt att en person ska komma åt.

## 3.2 Existerande IT-hot

En enkätundersökning har genomförts i syfte att hitta olika IT-hot samt påbörja för att få ihop data för en riskanalys, Frågorna finns i bilaga B.

Denna enkät skickades ut till personer som rekommenderades av tilldelad handledare på Bolagsverket.

Utöver enkätundersökningen med berörd personal har generellt populära hot analyserats och täckts av i denna undersökning.

## 3.3 Riskanalys

Riskanalysens syfte är att skapa en bild över risken som finns med de olika hoten. Detta har även givit en prioritering av hoten och har använts för att prioritera de olika hoten som tagits fram genom enkäten.

Genom en diskussion med en anställd som är med och arbetar med IT-säkerheten på Bolagsverket togs risken fram för de olika hoten.

Risken beräknas genom att sätt en siffra på konsekvensen och sannolikheten på hot, skalan på de båda värdena varierar från ett till fyra, där fyra är allvarlig och ett är försumbar. Sedan togs summan av dessa värden för vardera hot, denna summa är då värdet på risken för de individuella hoten.

Värdena sattes in i risk analys mall som syns i bilaga C.

## 3.4 Gap-Analys

Handlar om att ta reda på hur säkerheten ser ut i dagsläget för en specifik risk och jämför med hur den önskade säkerheten för denna risk.

Genom intervju med driftansvarig för applikationsservern framkom hur situationen ser ut i nuvarande läge.

Detta gjordes enbart på det valda hotet.

## 3.5 Framtagande av skyddsåtgärd

Genom en litteraturstudie på det valda hotet införskaffas data om hotet. Från den framtagna informationen från litteraturstudien tas det fram en eventuell skyddsåtgärd.

Den framtagna skyddsåtgärden ska då testas praktiskt och verifieras att den fungerar som önskas.

### **3.6 Test och verifikation av skyddsåtgärd**

För att vara säker på att en skyddsåtgärd fungerar ska den testas och verifieras.

För att veta hur testerna ska göras krävs först en litteraturstudie som innefattar kunskap för att veta hur den ska testas. Genom test ska det vara möjligt att se om skyddsåtgärden fungerar.

Genom avkodningsverktyget JD testas och verifieras den första delen av skyddsåtgärden och genom verifieringsverktyg testas den andra delen, skyddsåtgärden för jar filer.

## 4 Resultat

I detta kapitel presenteras resultaten av de olika undersökningarna. I underkapitel 4.1 visas resultatet från metoden Verksamhetsanalys, underkapitel 4.2 undersökningssvaren från metoden Existerande IT-hot, underkapitel 4.3 innehåller resultatet från metoden Riskanalys, underkapitel 4.4 innefattar resultatet från metoden Gap-Analys, underkapitel 4.5 innefattar resultat från metoden Framtagande av skyddsåtgärd och underkapitel 4.6 består av resultat från metod Test och verifikation av skyddsåtgärd.

### 4.1 Verksamhetsanalys

Svaren till intervjun för Verksamhetsanalysen visas nedan i Tabell 1. Direktattack är något som är direkt riktat mot Bolagsverket, bred attack är någon typ attack som riktas mot alla datorer det vill säga ej specifikt Bolagsverket.

Tabell 1: Svar på intervju frågorna i bilaga A.

Fråga nr	Svar
1	Ja, det finns och går att hitta på intranätet.
2	Ja, Stefan Ellström
3	<b>Borttagning på grund av sekretess.</b>
4	<b>Borttagning på grund av sekretess.</b>
5	En gång per år.
6	Överbelastningsattack, shellshock och virus.
7	Andra typer av hot: dataförlust (exempel en databas förloras), hårdvaruproblem (server går sönder/förstörs), data ändras och behörighetshantering (Person kommer åt data den ej har behörighet för).

Vid egen studie av blanketterna som fås in och ges ut till/av Bolagsverket framkom det att årsredovisningarna har en behandlingstid innan de publiceras för allmänheten. Detta gör att årsredovisningarna är intressant mål för att exempelvis förutspå aktiekurser.

### 4.2 Existerande IT-hot

De svar som kom in syns i tabellerna nedan.

**Tabell 2:** Svar på frågorna i bilaga B från person 1 och 2.

Fråga nr	Person 1	Person 2
1	Information om vår tekniska infrastruktur för att kunna utnyttja brister för att sänka, ta ner eller överbelasta våra tjänster. Servernamn, Serverversioner, patchnivåer, adminkonton osv. <b>Borttagning på grund av sekretess.</b>	<b>Borttagning på grund av sekretess.</b>
2	<b>Borttagning på grund av sekretess.</b>	<b>Borttagning på grund av sekretess.</b>
3	Största hotet tror jag är överbelastningsattacker.	Intrång och manipulering av data. Information som skulle vara intressant för "fel" personer. Belastningsattacker. Blockera våra tjänster.
4	Produktionsbortfall, avbrott och störningar för kunder, både slutkonsument och återförsäljare av vår data.	Manipulering av data för att kapa företag får stora konsekvenser för företagaren som drabbas. Belastning påverkar mest våra kunder.
5	Det känns som att terror i olika former blir vanligare och vanligare. Det verkar räcka med att det släpps en film med kontroversiellt innehåll eller att någon yttrar sig i något ämne så kommer grupperingar att vilja protestera genom att överbelasta servrar osv.	Relativt, mindre att manipulera data än belastning.
6	Vill inte svara på.	Svar saknas.
7	Svar saknas.	Svar saknas.

**Tabell 3:** Svar på frågorna i bilaga B från person 3 och 4.

Fråga nr	Person 3	Person 4
1	Inloggningsuppgifter för vidare attack. <b>Borttagning på grund av sekretess.</b> Ekonomisk information och hanterandet av betalningsflöde såsom fakturering och löneutbetalningar.	Information som kan hjälpa för att ta över företag eller personuppgifter.
2	Uppgifter så som utvecklingssamtal, eventuella känsliga uppgifter om hälsotillstånd och de sekretessbelagda.	Bankkontouppgifter, om de finns lagrade, eller företagsuppgifter för att kunna ta över företag.
3	<b>Borttagning på grund av sekretess.</b>	Belastningsattacker.
4	Felkonfigurationer eller arv med inställningar för kompatibilitet med äldre versioner av system, dessa äldre versioner blir inte alltid avslagna när de inte längre används. Brister i behörigheter eller för höga behörigheter på användare och mer. <b>Borttagning på grund av sekretess.</b> Har DDOS tjänst av ISP så det borde inte behövas hanteras för närvarandet..	Bolagsverkets tjänster ligger nere.
5	AV hittar kontinuerligt olika former av skadlig kod. Under 2015 har arbetsstationer kontaktat kända botnets.	Troligen låg.
6	Vill att det ska ske muntligt.	Säkra Java koden som används med kryptering.

7	Svar saknas.	Ingen aning.
---	--------------	--------------

**Tabell 4:** Svar på frågorna i bilaga B från person 5.

Fråga nr	Person 5
1	Ingen aning.
2	Ingen aning.
3	Ingen aning.
4	Ingen aning.
5	Liten pga eftersaknad av "värdefull" information.
6	<b>Borttagning på grund av sekretess.</b> Svaret handlade mycket om internsäkerhet.
7	Svar saknas.

De generella hoten utöver enkätundersökningen som tagits fram finns i tabell 5 nedan.

**Tabell 5:** lista de generella hoten som inte kom med i enkät undersökningen.

Generella hot	
Webbsidor (med skadligt innehåll) [13]	Operativ System (inte blivit uppdaterade i tid) [14]
Wifi [15]	Fysik installation av skadlig kod. [16]

### 4.3 Riskanalys

Riskanalysen handlar om att gå igenom de framtagna hoten från existerande IT-hot. Detta gjordes genom diskussion med IT-säkerhetsperson som är insatt i Bolagsverkets datorsäkerhet.

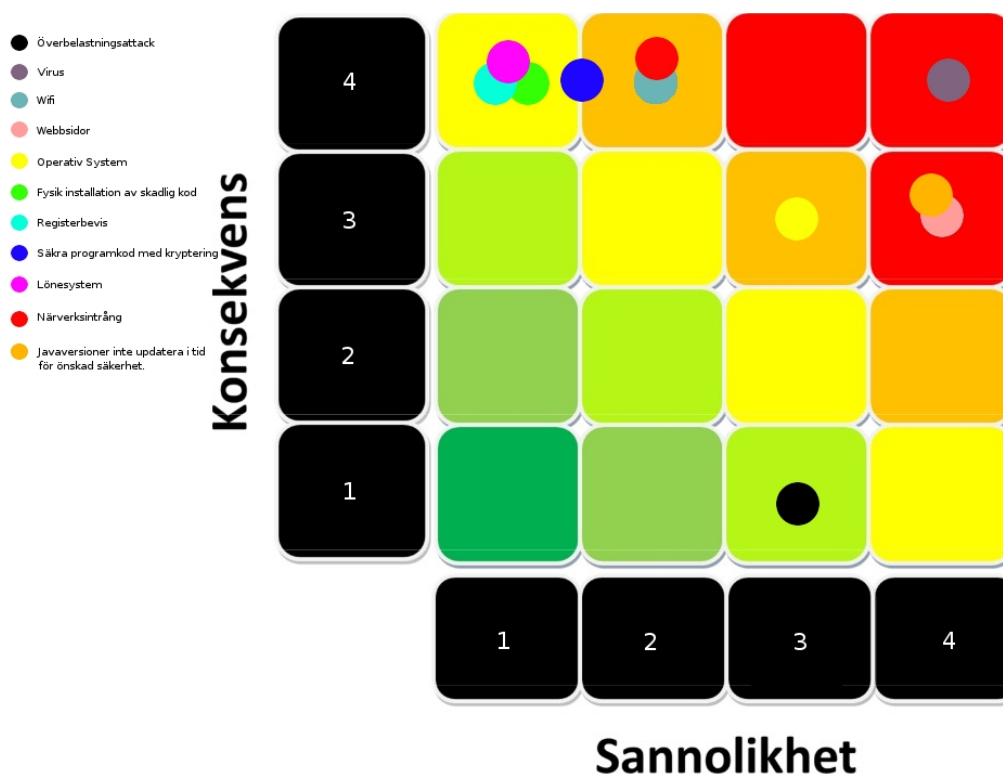
Det resultat som framkom syns nedan i tabell 6.

**Tabell 6:** Lista resultat från risk diskussion med säkerhetsinsatt anställd.

Hot	Konsekvens	Sannolikhet	Risk
Överbelastningsattack	1	3	4
Virus	4	4	8
Wifi	4	2	6
Webbsidor	3	4	7
Operativ System	3	3	6
Fysik installation av skadlig kod	4	1	5
Registerbevis	4	1	5

Säkra programkoden med kryptering	4	1.5	5.5
Lönesystem	4	1	5
Nätverksintrång	4	2	6
Javaversioner inte uppdateras i tid för önskad säkerhet	3	4	7

Från risk kolumnen i Tabell 6 ovan gjordes figur 6 nedan för en bättre översikt.



**Figur 1:** Visar ett diagram utifrån tabell 6 värden. Där de olika prickarna representerar de olika hoten (4 allvarligt/hög – 1 försumbar/låg).

Genom riskanalysen och samtal med handledare valdes risken ”säkra programkoden med kryptering” för att gå vidare med.

#### 4.4 Gap-analys

Genom intervju med applikationsserveransvarig togs det fram vad den nuvarande säkerheten ligger på för nivå för den valda risken i nuläget. De svar som gavs syns i tabell 7 nedan.

**Tabell 7:** Svar på intervjun för gap-analysen frågorna finns i bilaga D.

Fråga nr	Svar
1	Bolagsverket använder en JBoss "Enterprise Application Platform" (EAP) 6.4
2	Nej, i nuvarande situation används ingen kryptering på programkoden som finns på applikations servern.
2.1	Inget svar på grund av svaret på fråga två.
2.2	Inget svar på grund av svaret på fråga två.
3	Osäker men tror det.
3.1	Vet ej.

## 4.5 Framtagande av skyddsåtgärd

I underkapitel 5.5.1 är resultatet för att kryptera jar filer. Underkapitel 5.5.2 är resultatet för att kryptera applikationsserver paketen.

### 4.5.1 Skyddsåtgärd på jar filer

För att göra en kryptering på .jar filer kommer en klassladdare behövas. Vilket fungerar för att skydda en ickekörande kod från att bli avkodad via en känd avkodare som JD. Detta är ingen lösning för att skydda koden då klassladdaren måste leverera dess klassdefinitioner till "Java Virtual Machine" (JVM) via ett väldefinierat "Application Program Interface" (API). Klassladdarens definitions-klass har en rad med överlagringsmetoder för definitionsklassen, men alla dem kallas in i "defineClass(String, byte[], int, int, ProtectionDomain)" metoden. Det är "final" metod som kallar till JVM:en. Där används en metod skriven i ett annat språk efter det att några kontroller genomförts. Ingen klassladdare kan undvika att kalla på denna metod. [17]

Detta är den enda metoden som kan skapa en klass av en byte vektor, denna byte vektor måste innehålla klartexten av klassdefinitionen i ett väl dokumenterat format. På grund av detta är det nu en enkel avlyssning på alla kallelser till/från denna metod, genom avlyssningen fås då alla okrypterade klasser som går att avkoda med en avkodare som JD. Denna avlyssning görs genom att modifiera metoden "defineClass(String, byte[], int, int, ProtectionDomain)" att logga lite mer när den körs. [17]

För att skydda .jar samt göra det svårare att utföra dekompilering så är det bättre att använda ett komplicerande-verktyg (som gör koden svår läst). [18] Verktyget som har används under projektet är Proguard. Proguard kördes med följande inställningsfilen som syns i tabell 8 nedan.

**Tabell 8:** Är inställningsfilen som säger vilka inställningar som Proguard ska köras med.

Proguard konfigurationsfil
<pre>-injars Statistik.jar -outjars done.jar -libraryjars &lt;java.home&gt;/lib/rt.jar  -optimizationpasses 3 -overloadaggressively -repackageclasses '' -allowaccessmodification  -keep public class statistik.Statistik {     public static void main(java.lang.String[]); }</pre>

Flaggan ”-injars” vilken jar vill som Proguard ska köras på och ”-outjars” vad filen ska heta när Proguard är klart samt ”-libraryjars” vart den kan finna rt.jar (rt.jar följer med vid installation av Java SE och EE). ”-optimizationpasses 3” berättar för komplicerande-verktyget att den kan köra upp till tre optimeringsförsök (det är max vad Proguard kan), ”-overloadaggressively” betyder att verktyget ska göra koden så svårläst som den kan och ”-repackageclasses '' ” säger att Proguard kommer ompaketera alla klassfiler som får ett nytt namn samt ”-allowaccessmodification” som kan ge en bättre optimering. Flaggan ”-keep” används på metoder som inte får förändras till exempel main och i vissa fall andra metoder.

För att kontrollera att koden är oförändrad i jar filen användes verktyget Jarsigner vilket signerar koden, denna signatur används för att kontrollera att koden ej blivit manipulerad. [19] Jarsigner vill också ha en plats att förvara och få signerings nycklarna, för det så användes verktyget keytool. [9]

Keytool delen ska vara uppsatt innan Jarsigner kunde köras. För att sätta upp Keytool kördes terminal kommandot som visas i tabell 9.

**Tabell 9:** Visar kommandot som kördes för att sätta upp keytool databas för tester med genererade nycklar.

Keytool kommando
<pre>keytool -genkeypair -dname "cn=gunnar kjällkod, ou=test, o=bolags, c=SV" -alias signFile -keypass bolag2 -keystore testKeystore -storepass bolag1 -keyalg RSA -validity 365</pre>

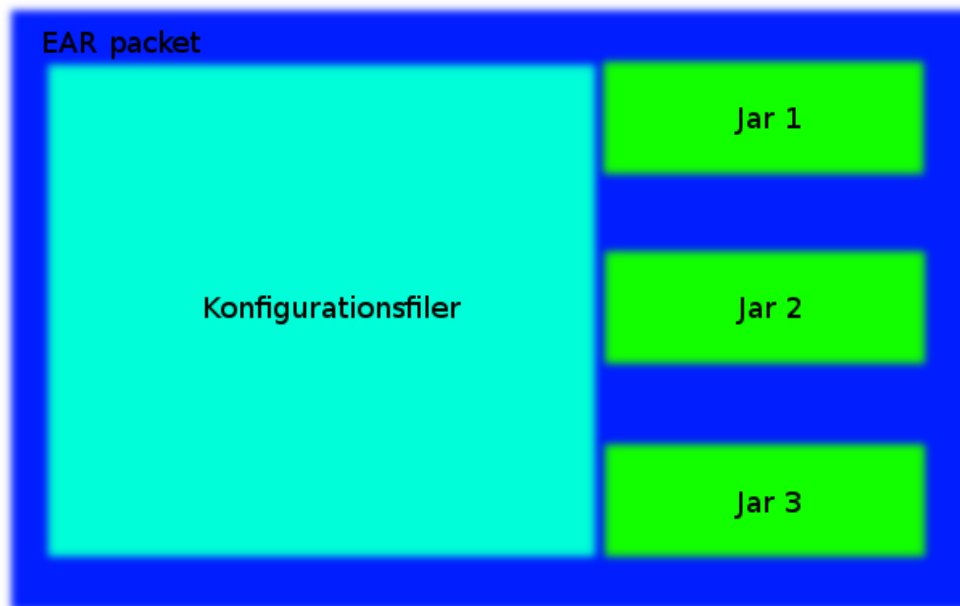
För att signera jar filen så kördes ett kommando i terminalen som finns nedan i tabell 10.

**Tabell 10:** Visar terminal kommandot som kördes för att signera jar test filen.

Jarsigner kommando
<pre>jarsigner -keystore testKeystore -signedjar result.jar done.jar signFile</pre>

#### 4.5.2 Skyddsåtgärder på JBoss EAP

JBoss packeten som används på Bolagsverket är .EAR som fungerar enligt figur 2 nedan.



**Figur 2:** Förklarande bild för .EAR packet som används i JBoss, kan innehålla mer än tre st jar filer.

Konfigurationsfilerna i EAR-paket innehåller inställningar för hur JBoss ska arbeta med paket, de innehåller något som heter "secure-domains" som säger vart JBoss kan hitta säkerhetsmetoder för det paketet, vilket undersökes vidare för att försöka hitta lösningen där. [20]

Kryptera EAR-paketet (eller WAR-paket) kommer inkludera samma problem som JAR-paketet, ty ingen mening att kryptera EAR-paketet, förklaring för detta står i början av kapitel 5.5.1. [21]

"Secure-domains" är säkerhetshanteringen för EAR filer. "Secure-domains" är en eller flera "secure-domain" som är grundelementet i "secure-domains". I "secure-domain" de olika inställningarna görs via XML kod. De olika delarna i "secure-domain" förklaras i listan nedan. [20]

1. Namn på "secure-domain":en, namn på en "secure-domain" som ska förlängas med denna och vilken hur den ska använda cache (Map baserat eller infinispän). [22]
2. Autentisering – som innehåller en lista av inloggningsmoduler som kan vara hela autentiseringsklassen eller en av de förkortade namnen i bilaga G under rubrik Autentisering, där finns även kort XML exempel. [22]

3. Autentisering Java Authentication Service Provider Interface (JASPI) - Detta är alternativ till punkt 2 i listan. [22] XML exempel finns i bilaga G under rubriken autentisering-JASPI.
4. Tillstånd – Handlar om åtkomst och/eller nekad åtkomst som bestäms med ett modulärt system. [22] De olika modulerna som finns och XML exempel finns i bilaga G under rubriken tillstånd.
5. Kartläggning – detta alternativ har ytterligare kartläggning av principer, autentiseringsuppgifter, roller och attribut för ämnet. [22] Strukturen och olika specifika alternativen finns i bilaga G under rubriken Kartläggning.
6. Granskning – Granskningselementet kan användas för anpassade granskningstillhandahållare. [22] XML-exempel finns i bilaga G under rubriken Granskning.
7. Java Secure Socket Extension (JSSE) – definierar konfigurationen för nyckellagring och pålitlig-lagring som kan användas för SSL konfiguration eller certifikatlagring eller hämtning. [22] För XML exempel och de olika attributen, se bilaga G under rubriken JSSE.

## 4.6 Test och verifikation av skyddsåtgärd

Vid tidsbrist, och på grund av att det inte var initialhotet, har ingen implementation eller test skett av "secure-domains".

I underkapitlet 4.5.1 förklaras varför kryptering inte testats.

Proguard test, för att se koden innan komplicerade-verktyget används se bilaga E. Proguard inställningar finns i tabell 8 som ligger i underkapitel 4.5.1. Koden efter komplicerade-verktyget kört finns i bilaga F. Den koden har tagits fram från den nya jar-filen med hjälp av avkodningsverktyget JD.

För att verifiera att koden har blivit signerad användes Jarsigner med flagorna "-verify" och "-verbose". Detta är den rekommenderade veriferingen av Oracle. Utskriften från verifikationen ses nedan i tabell 11.

**Tabell 11:** Visar ett verifierings test utdata.

```
Verifierings kommando

$ jarsigner -verify -verbose done.jar

s          714 Tue May 12 14:32:24 CEST 2015 META-INF/MANIFEST.MF
s          734 Tue May 12 14:32:24 CEST 2015 META-INF/TEST.SF
m         1032 Tue May 12 14:32:24 CEST 2015 META-INF/TEST.DSA
sm          74 Tue May 12 12:54:22 CEST 2015 statistik/README.md
sm          390 Tue May 12 12:54:22 CEST 2015 b.class
sm         3123 Tue May 12 12:54:22 CEST 2015 a.class
sm          390 Tue May 12 12:54:22 CEST 2015 d.class
sm         3274 Tue May 12 12:54:22 CEST 2015 c.class
sm         1415 Tue May 12 12:54:22 CEST 2015 statistik/Statistik.class

s = signature was verified
m = entry is listed in manifest
k = at least one certificate was found in keystore
i = at least one certificate was found in identity scope

jar verified
```

Enligt verifieringsverktyg i Jarsigner är koden nu signerad.

Exempel på hur det ser ut när koden inte är signerad syns i tabell 12 nedan.

**Tabell 12:** Visar verifierings utskrift försök på osignerad kod.

```
Verifering av ej signerad kod

$ jarsigner -verify -verbose Statistik.jar

          0 Tue May 05 14:15:14 CEST 2015 META-INF/
          0 Tue May 05 14:15:14 CEST 2015 statistik/
          74 Tue May 05 14:15:14 CEST 2015 statistik/README.md
          846 Tue May 05 14:15:14 CEST 2015 statistik/RandomTextPrint$1.class
         6494 Tue May 05 14:15:14 CEST 2015 statistik/RandomTextPrint.class
          826 Tue May 05 14:15:14 CEST 2015 statistik/StatLetter$1.class
         5477 Tue May 05 14:15:14 CEST 2015 statistik/StatLetter.class
         1243 Tue May 05 14:15:14 CEST 2015 statistik/Statistik.class
          210 Tue May 05 14:15:12 CEST 2015 META-INF/MANIFEST.MF

s = signature was verified
m = entry is listed in manifest
k = at least one certificate was found in keystore
i = at least one certificate was found in identity scope
```

## 5 Diskussion

Detta kapitel tar upp projektgenomförarens egna tankar kring resultatet och projektet.

Underkapitel 5.1 innehåller diskussionen för Verksamhetsanalysen, underkapitel 5.2 tar upp slutsatser från Existerande IT-hot, underkapitel 5.3 tar fram slutsatser för Riskanalysen, underkapitel 5.4 innehåller slutsatser om Gap-analysen, underkapitel 5.5 beskriver slutsatserna för Framtagande av skyddsåtgärderna, underkapitel 5.6 nämner slutsatser från Test och verifikation av skyddsåtgärd, underkapitel 5.7 är Förutsatta mål i projektet och underkapitel 5.9 handlar om Etiska aspekter.

### 5.1 Verksamhetsanalys

Det blev inte så mycket text i rapporten, men det var den del som var absolut störst i projektet, och på grund av detta har den lämnat intrycket om att eventuellt något missades i undersökningen. Som ensam undersökare blev det en stor chock över mängden blanketter som faktiskt används för att rapportera till svenska myndigheter som organisation.

Det som togs med från undersökningarna var att Bolagsverket känner sig relativt ohotat (gällande direkt riktade hot) på grund av att myndigheten inte ses som en nyckelmyndighet för att Sverige ska fungera, enligt Bolagsverket personal.

Bra saker Bolagsverket gör för att öka säkerheten är att viss personal har möten en gång per år där de går igenom vilken information som hanteras och utför riskanalyser.

### 5.2 Existerande IT-hot

Enkäten gav tyvärr ingen röd tråd som önskades av författaren vilket ger möjligheten att eventuella hot missats och det är på grund av detta som intervjun framkom i riskanalysen.

Varför det inte framgick en röd tråd i enkäten kan bero på tre saker. Den första anledningen var att enkäten skickades ut till för få antal personer. Den andra är att det bara var personer som arbetar med IT-delen inom Bolagsverket och troligen är några av dem med på den årliga säkerhetsundersökningarna som nämns i kapitel 5.1 ovan. Den tredje anledningen är att frågorna kanske inte var tillräckligt öppna/specifika för få fram en röd tråd som visar vad som är odiskutabelt det hot som måste tas i tur med.

### 5.3 Riskanalys

Det som saknades i riskanalysen var att den egentligen ska genomföras i en grupp som diskuterar vardera hot för att på så sätt få mer åsikter och förhoppningsvis få ut verkligare konsekvens- och sannolikhetsvärderingar. Under detta projekt var det bara två personer, och av dessa två bara en, som arbetar på Bolagsverket och har en sann inblick i organisationen.

Efter riskanalysen vid diskussion med handledare på Bolagsverket valdes risken för okrypterad programkod, vilket inte var den största risken. Det fanns tre anledningar för detta, den första anledningen var att risken som väljs måste vara genomförbar och den andra anledningen var att det fanns en tidsbrist då det måste vara genomförbart på en viss tid samt anledning tre, som var att hotet var tvunget att vara datatekniskt för examens ämnet. Därav blev valet risken för okrypterad programkod.

### 5.4 Gap-analys

Frågorna för bilaga D blev dessa frågor för att vid intervjuer och småprat var det ingen som trodde, alternativt var osäkra, om det fanns någon kryptering på koden, så steg ett var att kontrollera hur det låg till och vilka möjligheter som fanns.

Denna analys handlar om att få en bild om den nuvarande säkerhetsinsatsen på en viss risk och jämföra med den önskade. Därför blir frågorna anpassade efter risken i fråga. Då misstanken redan innan var att det inte fanns någon kryptering på den valda risken blev frågorna som de visas i bilaga D.

### 5.5 Framtagande av skyddsåtgärd

Den framtagna skyddsåtgärden komplicerande-verktyg är något som myndigheten inte använder idag, men är den bättre vägen för att göra det svårare att manipulera koden då kryptering som nämns i resultat är en dålig och osäker lösning i detta fall.

Signering av kod är ett sätt att kunna göra två saker beroende på vad tanken är med signaturen, den kan användas av en organisation på så sätt att säga denna kod är något som organisationen stödjer/gillar/”litar på” och signerar av på, det andra sättet är för att kontrollera att koden inte blivit manipulerad och ser likadan ut som när den signerades.

JBoss har många säkerhetsinställningar, det finns flera dokument på antal hundra sidor på JBoss webbplats som handlar om säkerhet och säkerhetsinställningar. Men det enda som hittades angående kryptering var ”data-source” (heter ibland också ”vault” i JBoss dokumentationen), vilket är vad JBoss kallar deras lagring för databaskopplingars lösenord och användare. ”Data-source” lagring går att kryptera via JBoss, vilket betyder att applikationsserven har hand om kryptering och dekryptering av dessa lagringar.

## 5.6 Test och verifikation av skyddsåtgärd

Det finns en besvikelse av att tidsbristen åstadkom att implementation av "secure-domains" på JBoss Enterprise Application Platform (EAP) inte genomfördes, samtidigt hade det blivit mycket arbete på sidan som att installera en test JBoss EAP server på laptopen som ska ha samma inställningar som Bolagsverket har på den som körs för deras Java kod i produktion. Det hade tagit tid att ta reda på inställningarna, samt mycket läsning för kunna verifiera och testa av inställningarna på "secure-domains", vilket hade slutat i en stor litteraturstudie för att se hur det skulle gå att testa på ett verkligt sätt. Sedan skulle det också krävas en litteraturstudie för att förstå sig på alla de olika delarna i "secure-domain" på tillräcklig nivå för att få en bra bas att testa mot, så det inte blir något halvgjort.

Jarsigner är vad Oracle använder i sina signeringsexempel och rekommenderar, vilket gör att det är vettigt att använda i implementation för test och verifisering. Testerna visade att koden blev signerad och att det gick att kontrollera. Går även att göras på så sätt att koden verifieras mot en server externt och internt.

Proguard gav en trygghet att använda då mjukvaran har uttalande från människor och företag som IBM och M.B., Statestep för kvalitet och robusthet. Problemet som fortfarande finns och syns, även i den komplicerade-koden i bilaga F, är att det fortfarande går att återställa till koden igen, men att den är betydligt mycket svårare att läsa och förstå. Med tid och envishet går koden att förstå och manipulera så det inte är ett skydd utan mer ett verktyg som gör det svårare utan att påverka kodens funktionalitet.

## 5.7 Förutsatta mål

De Konkreta och verifierbara målen kommer nu ses över för att kontrollera att det uppfylls och blivit av klarade.

1. Mål 1 kartläggning av eventuella hot, där vardera hot ska få en sammanfattning som ska innehålla de risker som hotet har, på övergripande nivå. - Ja (se bilaga H).
2. Mål 2 diskutera och prioritera för att välja hot att arbeta vidare med, svaret finns i resultatet, se underkapitel 4.3 där riskanalysens riskvärde är prioriteringen, diskussionen skedde med handledare Martin Bylund och nämns i samma underkapitel.
3. Mål 3 samtliga skyddsåtgärder ska dokumenteras och beskrivas – vilket gjorts i resultatet, se underkapitel 3.5.
4. Mål 4 att skyddsåtgärden som tagits fram ska testas och verifieras praktiskt. – Ja och nej, tyvärr tog tiden slut för att hinna testa och verifiera JBoss "secure-domains" som nämns i underkapitel 5.6. Resultatet för tester och verifiseringar finns i underkapitlet 4.6.

## 5.8 Framtida forskning

Framtida punkter som kan utforskas är dels de hoten som valdes bort, men även på det valda hotet.

Vid fortsättning inom det valda hotet skulle det vara möjligt att spinna vidare på "secure-domains" och eventuella andra skydd som finns i JBoss för att skydda från manipulering.

Om istället en fortsatt forskning sker inom de hot som valts bort måste det börjas med en gap-analys. För att sedan se vilka möjliga skyddsåtgärder som finns och till sist utföra tester för att verifiera att dessa skyddsåtgärden fungerar som det är tänkt.

## 5.9 Etiska aspekter

De etiska aspekterna för projektet är att minska risken att någon lyckas utföra olagliga handlingar genom att till exempel ändra ägare på bolag eller förfälska handlingar. Med en mindre risk ökar allmänheten och organisationers förtroende för myndigheten Bolagsverket. Vilket ger en positiv bild mot svenska myndigheter, då människor känner att de kan lita på landets myndigheter.

Genom att skydda data från manipulation och från bort fall så skyddas organisationer från att någon kapar dessa och gör något med organisationen som inte är önskvärt.

## Källförteckning

- [1] Informationssäkerhet.se, "Vad är informationssäkerhet?"  
<https://www.informationssakerhet.se/sv/informationssakerhet/allmant/>  
Hämtad 2015-06-01
- [2] Oracle Java SE Documentation, "Class ClassLoader"  
<https://docs.oracle.com/javase/7/docs/api/java/lang/ClassLoader.html>  
Hämtad 2015-04-29
- [3] Java Decompiler, "JD Project"  
<http://jd.benow.ca/> Hämtad 2015-05-27
- [4] Excelsior Library, "Protect Your Java Code – Through Obfuscators And Beyond"  
<http://www.excelsior-usa.com/articles/java-obfuscators.html> Publicerad 2015-02-20. Hämtad 2015-04-28
- [5] Proguard, "Main"  
<http://proguard.sourceforge.net> Hämtad 2015-04-30
- [6] Proguard, "Testimonial"  
<http://proguard.sourceforge.net/testimonials.html> Hämtad 2015-04-30
- [7] Proguard, "Results"  
<http://proguard.sourceforge.net/results.html> Hämtad 2015-06-15
- [8] Oracle Java SE Documentation, "jarsigner – JAR Signing and Verification tool"  
<http://docs.oracle.com/javase/6/docs/technotes/tools/windows/jarsigner.html> Hämtad 2015-05-27
- [9] Oracle Java SE Documentation, "keytool – Key and Certificate Management Tool"  
<http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html> Hämtad 2015-05-26
- [10] JBoss Developer, "Overview"  
<http://www.jboss.org/products/eap/overview/> Hämtad 2015-05-27

- [11] Myndigheten för samhällsskydd och beredskap, "Välkommen till MSB!"  
<https://msb.se/> Hämtad 2015-06-15
- [12] Informationssäkerhet.se, "Metodstöd"  
<https://www.informationssakerhet.se/sv/Metodstod/Metodstod/> Hämtad 2015-02-10
- [13] BeEF, "What is BeEf?"  
<http://beefproject.com/> Hämtad 2015-04-10
- [14] DinSäkerhet.se, "Skydda din dator"  
<http://www.dinsakerhet.se/Informationssakerhet/Konkreta-rad-informationssakerhet/It-sakerhet/> Publicerad 2013-01-25 Hämtad 2015-05-14
- [15] eSecurity Planet, "Top ten Wi-Fi Security Threats"  
<http://www.esecurityplanet.com/views/article.php/3869221/Top-Ten-WiFi-Security-Threats.htm> Publicerad 2010-03-08 Hämtad 2015-04-26
- [16] IEEE SPECTRUM, "The Real Story of Stuxnet"  
<http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet>  
Publicerad 2013-02-26 Hämtad 2015-04-26
- [17] JavaWorld, "Cracking Java byte-code encryption"  
<http://www.javaworld.com/article/2077342/core-java/cracking-java-byte-code-encryption.html> Publicerad 2003-05-09 Hämtad 2015-05-03
- [18] Proguard, "Main"  
<http://proguard.sourceforge.net/main.html> Hämtad 2015-07-12
- [19] Oracle Java Documentation, "Signing and Verifying jar Files"  
<https://docs.oracle.com/javase/tutorial/deployment/jar/signindex.html>  
Hämtad 2015-04-30
- [20] Redhat, "JBoss Enterprise Application Platform 6.1 – Security Guide"  
[https://access.redhat.com/documentation/en-US/JBoss\\_Enterprise\\_Application\\_Platform/6.1/html-single/Security\\_Guide/index.html](https://access.redhat.com/documentation/en-US/JBoss_Enterprise_Application_Platform/6.1/html-single/Security_Guide/index.html) Hämtad 2015-04-30
- [21] Stackoverflow, "Encrypting War files"  
<http://stackoverflow.com/questions/1175008/encrypting-war-files>  
Publicerad 2009-07-24 Hämtad 2015-05-10
- [22] docs.jboss.org, "Security subsystem configuration"  
[https://docs.jboss.org/author/display/AS7/Security+subsystem+configuration?\\_sscc=t](https://docs.jboss.org/author/display/AS7/Security+subsystem+configuration?_sscc=t) Publicerad 2012-11-20 Hämtad 2015-05-26

## **Bilaga A: verksamhetsanalys intervju frågor**

Fråga 1: Finns det någon policy för informationssäkerhet?

Fråga 2: Finns det en utsatt samordnare för informationssäkerhet? Om ja, vad heter personen?

Fråga 3: Hur ser engagemanget ut för datasäkerheten? (1-10) skala där 10 är bra och 1 dålig.

Fråga 4: Vad är de ute efter när de attackera Bolagsverket?

Fråga 5: Hur ofta görs en ny informationsundersökning?

Fråga 6: Vilka attacker har ni historiskt blivit utsatta för?

Fråga 7: Övrigt.

## **Bilaga B: enkät undersökning för IT-hot**

### **Bakgrund**

För att kunna känna tillit till myndigheter är det viktigt att företag och allmänheten känner att den information som myndigheterna har tillgång till inte kommer att hamna i fel händer. Detta gör att informationssäkerheten är en viktig del i arbetet.

### **Orsak till enkät**

Jag gör ett examensarbete om informationssäkerhet på Bolagsverket och samlar in data till min förundersökning. Denna förundersökning kommer vara en av grunderna för min riskanalys.

### **Ansvarig utgivare:**

Jag heter Mikael Falck och är universitetsstuderande på Mittuniversitet i Sundsvall med inriktning Civilingenjör i datateknik. Jag gör just nu min kandidatexamensarbete som handlar om informationssäkerhet på Bolagsverket.

Vid eventuella frågor nås jag på telefon: 072 578 32 09 eller via E-post:

[Mikael.Falck@bolagsverket.se](mailto:Mikael.Falck@bolagsverket.se) eller [mifa1100@student.miun.se](mailto:mifa1100@student.miun.se)

Mina handledare på Bolagsverket är Martin Bylund (Sektionschef och it-arkitekt) och Mikael Larsson Österström (teknisk testare).

**Vid dataintrång, vilken information är det störst sannolikhet att de är ute efter enligt dig? (gärna en motivering)**

**Vid dataintrång, vilken information, som finns på Bolagsverket, har enligt dig störst konsekvens om den skulle komma ut? (gärna en motivering)**

**Vilka IT-hot finns det mot Bolagsverket enligt dig? (gärna en motivering)**

**Vad har de olika IT-hoten för konsekvenser enligt dig?**

**Vad har de olika IT-hoten för sannolikhet att inträffa enligt dig?**

**Är det något inom säkerheten på Bolagsverket som saknas enligt dig?**

**Är det någon fråga som saknas skriv gärna här:**

## Bilaga C: exempel mall på risk dokument för ett individuellt hot

Här under är mallen för riskanalys en sådan sida per individuellt hot.

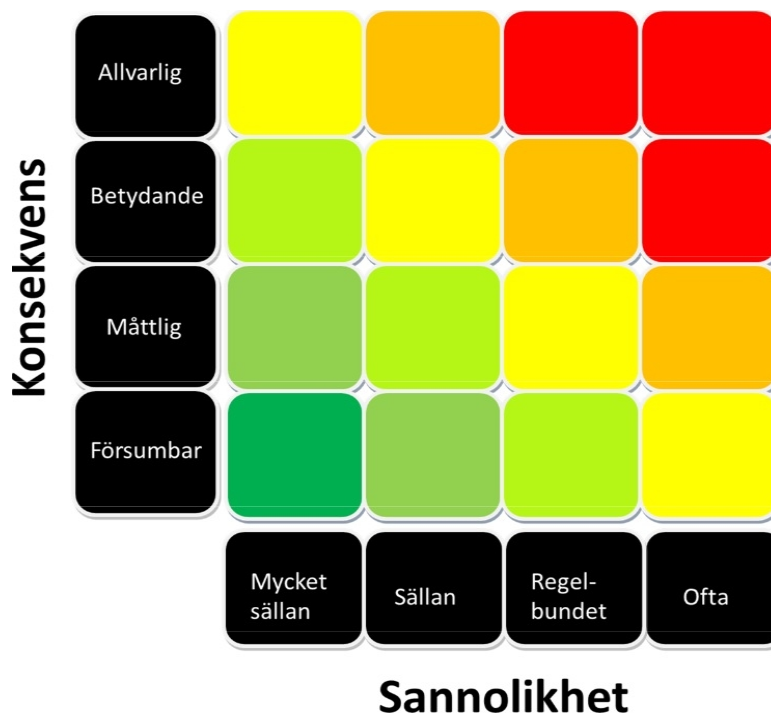
### Rubrik (hotets namn)

Eventuell för med svar.

### Ranking mellan 1-4

Konsekvens	Sannolikhet
Siffervärde på konsekvensen av hotet om det inträffar.	Siffervärde på sannolikheten att hotet in träffar.

Konsekvens: förklaring
Här är en kort text som förklarar konsekvensen av hotet.



I figuren ovan ritas en prick in där platsen bestäms av siffervärdet. 4 är allvarlig /ofta och 1 är försumbar/mycket sällan.

## **Bilaga D: intervjufrågor för gap-analys**

1. Vad är det för applikationsserver som Bolagsverket använder?
2. Finns en kryptering för programkoden idag? Om ja, se underfrågorna.
  1. Vad är det för kryptering?
  2. Hur pass säker skulle du säga den är?
3. Finns det möjlighet för kryptering på programkoden i applikationsservern? Om ja, se underfrågan.
  1. Om ja, vilka alternativ för kryptering finns det?

## **Bilaga E: ursprungs kod innan komplicerade verktyg används**

koden går också att hämta på Mikael Falcks github:  
<https://github.com/mejan/Crypt> .

## Bilaga F: Programkod efter komplicerade-verktyget har körts på den

Här nedan syns den komplicerade koden då Proguard utfört sitt arbete på den.

```
public class Statistik
{
    public static void main(String[] paramArrayOfString)
    {
        Locale.setDefault(new Locale("sv"));
        (paramArrayOfString = new c()).a("/home/mejan/Documents/skola/VT-15/Kryptografi/Assignment/a1/svenskaAlpha/bibeln.txt");
        paramArrayOfString.b("/home/mejan/Documents/skola/VT-15/Kryptografi/Assignment/a1/svenskaAlpha/resultOneLetter.txt");
        String str = "/home/mejan/Documents/skola/VT-15/Kryptografi/Assignment/a1/svenskaAlpha/randomText.txt";
        paramArrayOfString = new a("/home/mejan/Documents/skola/VT-15/Kryptografi/Assignment/a1/svenskaAlpha/bibeln.txt");
        BufferedWriter localBufferedWriter = null;
        try
        {
            (localBufferedWriter = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(str), "utf-8"))).write(paramArrayOfString.a);
            try
            {
                localBufferedWriter.close();
                return;
            }
            catch (Exception localException1)
            {
                return;
            }
        }
        catch (IOException paramArrayOfString)
        {
            System.out.println(paramArrayOfString.toString());
            try
            {
                localBufferedWriter.close();
                return;
            }
            catch (Exception localException2)
            {
                return;
            }
        }
        finally
        {
            try
            {
                localBufferedWriter.close();
            }
            catch (Exception localException3)
            {
            }
        }
        throw paramArrayOfString;
    }
}

public final class a
{
    private Map jdField_a_of_type_JavaUtilMap = new HashMap();
    private int jdField_a_of_type_Int = 0;
    public String a;
    private Random jdField_a_of_type_JavaUtilRandom = new Random();

    public a(String paramString)
    {
        this.jdField_a_of_type_JavaLangString = "";
    }
}
```

```
a(paramString);
b();
}

private void a(String paramString)
{
    paramString = new BufferedReader(new InputStreamReader(new
FileInputStream(paramString), Charset.forName("UTF-8")));
    HashMap localHashMap = new HashMap();
    int i;
    while ((i = paramString.read()) != -1)
        if ((Character.isLetter(i = (char)Character.toLowerCase(i))) || (i ==
32))
            if (!localHashMap.containsKey(Character.valueOf(i)))
                {
                    localHashMap.put(Character.valueOf(i), Double.valueOf(1.0D));
                    this.jdField_a_of_type_Int += 1;
                }
            else
                {
                    Double localDouble =
Double.valueOf(((Double)localHashMap.get(Character.valueOf(i))).doubleValue() +
1.0D);
                    localHashMap.put(Character.valueOf(i), localDouble);
                    this.jdField_a_of_type_Int += 1;
                }
            this.jdField_a_of_type_JavaUtilMap = a((HashMap)localHashMap);
            a();
        }

private void a()
{
    Iterator localIterator =
this.jdField_a_of_type_JavaUtilMap.entrySet().iterator();
    while (localIterator.hasNext())
        {
            Map.Entry localEntry = (Map.Entry)localIterator.next();
            Double localDouble =
Double.valueOf(((Double)this.jdField_a_of_type_JavaUtilMap.get(localEntry.getKey())
).doubleValue() / this.jdField_a_of_type_Int);
            this.jdField_a_of_type_JavaUtilMap.put(localEntry.getKey(), localDouble);
        }

private static HashMap a(HashMap paramHashMap)
{
    Collections.sort(paramHashMap = new LinkedList(paramHashMap.entrySet()),
new b());
    LinkedHashMap localLinkedHashMap = new LinkedHashMap();
    paramHashMap = paramHashMap.iterator();
    while (paramHashMap.hasNext())
        {
            Map.Entry localEntry = (Map.Entry)paramHashMap.next();
            localLinkedHashMap.put(localEntry.getKey(), localEntry.getValue());
        }
    return localLinkedHashMap;
}

private String a()
{
    if (this.jdField_a_of_type_JavaLangString == null)
        return null;
    char[] arrayOfChar = this.jdField_a_of_type_JavaLangString.toCharArray();
    ArrayList localArrayList = new ArrayList(arrayOfChar.length);
    for (char c : arrayOfChar)
        localArrayList.add(Character.valueOf(c));
    Collections.shuffle(localArrayList, this.jdField_a_of_type_JavaUtilRandom);
    arrayOfChar = new char[localArrayList.size()];
    for (int i = 0; i < localArrayList.size(); i++)
        arrayOfChar[i] = ((Character)localArrayList.get(i)).charValue();
    return new String(arrayOfChar);
}

private void b()
{
    Iterator localIterator =
this.jdField_a_of_type_JavaUtilMap.entrySet().iterator();
    while (localIterator.hasNext())
        {
            Map.Entry localEntry;
            int i = (int)(((Double)localEntry
(Map.Entry)localIterator.next()).doubleValue() * 2000.0D);
            for (int j = 0; j < i; j++)
```

```
        this.jdField_a_of_type_JavaLangString += localEntry.getKey();
    }
    this.jdField_a_of_type_JavaLangString = a();
}

import java.util.Comparator;
import java.util.Map.Entry;

final class b
    implements Comparator
{
    public final int compare(Object paramObject1, Object paramObject2)
    {
        return ((Comparable)
((Map.Entry)paramObject2).getValue()).compareTo(((Map.Entry)paramObject1).getVa
lue());
    }
}

public final class c
{
    private Map jdField_a_of_type_JavaUtilMap = new HashMap();
    private int jdField_a_of_type_Int = 0;

    public final void a(String paramString)
    {
        this.jdField_a_of_type_JavaUtilMap.clear();
        BufferedReader localBufferedReader = new BufferedReader(new
InputStreamReader(new FileInputStream(paramString), Charset.forName("UTF-8")));
        String str = "";
        HashMap localHashMap = new HashMap();
        while ((paramString = localBufferedReader.read()) != -1)
            if ((Character.isLetter(paramString) =
(char)Character.toUpperCase(paramString)) || (paramString == 32))
            {
                str = str + paramString;
                if (!localHashMap.containsKey(str))
                    localHashMap.put(str, Integer.valueOf(1));
                else
                    localHashMap.put(str,
Integer.valueOf(localHashMap.get(str).hashCode() + 1));
                str = "";
                this.jdField_a_of_type_Int += 1;
            }
        this.jdField_a_of_type_JavaUtilMap = a((HashMap)localHashMap);
    }

    private static HashMap a(HashMap paramHashMap)
    {
        Collections.sort(paramHashMap = new LinkedList(paramHashMap.entrySet()),
new d());
        LinkedHashMap localLinkedHashMap = new LinkedHashMap();
        paramHashMap = paramHashMap.iterator();
        while (paramHashMap.hasNext())
        {
            Map.Entry localEntry = (Map.Entry)paramHashMap.next();
            localLinkedHashMap.put(localEntry.getKey(), localEntry.getValue());
        }
        return localLinkedHashMap;
    }

    public final void b(String paramString)
    {
        BufferedWriter localBufferedWriter = null;
        Iterator localIterator =
this.jdField_a_of_type_JavaUtilMap.entrySet().iterator();
        double d1;
        for (String str = ""; localIterator.hasNext(); str = str + d1 + "
Percent.\n")
        {
            Map.Entry localEntry = (Map.Entry)localIterator.next();
            str = str + (String)localEntry.getKey() + " ";
            double d2 = ((Integer)localEntry.getValue()).intValue() /
this.jdField_a_of_type_Int * 100.0D;
            long l = ()Math.pow(10.0D, 4.0D);
            d1 = Math.round(d2 * l) / l;
            str = str + " Har i decimalform: ";
        }
        try
        {
            (localBufferedWriter = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream(paramString), "utf-8"))).write(str);
        }
    }
}
```

```
        try
        {
            localBufferedWriter.close();
            return;
        }
        catch (Exception localException1)
        {
            return;
        }
    }
    catch (IOException localIOException)
    {
        System.out.println(localIOException.toString());
        try
        {
            localBufferedWriter.close();
            return;
        }
        catch (Exception localException2)
        {
            return;
        }
    }
    finally
    {
        try
        {
            localBufferedWriter.close();
        }
        catch (Exception localException3)
        {
        }
    }
    }
    throw paramString;
}
}

final class d
    implements Comparator
{
    public final int compare(Object paramObject1, Object paramObject2)
    {
        return ((Comparable)
((Map.Entry)paramObject2).getValue()).compareTo(((Map.Entry)paramObject1).getVa
lue());
    }
}
}
```

## Bilaga G: alternativa listor för inställningar i "secure-domain" för Jboss

Denna bilaga innehåller tabeller med alternativ för olika delar i "secure-domain" och korta XML exempel.

### Autentisering

XML exempel i tabell 1 nedan.

XML exempel
<pre>&lt;login-module code="..." flag="..."&gt;   &lt;module-option name="..." value="..." /&gt; &lt;/login-module&gt;</pre>

Tabell 1: XML exempel för autentiserings delen i "secure-domain" för EAR-paketet.

Tabell 2 nedan visar alternativ för inloggningsmodul i autentisering.

Enstaka inloggningsmoduler	
Kod	Klassnamn
Client	org.jboss.security.ClientLoginModule
Certificate	org.jboss.security.auth.spi.BaseCertLoginModule
CertificateUsers	org.jboss.security.auth.spi.BaseCertLoginModule
CertificateRoles	org.jboss.security.auth.spi.CertRolesLoginModule
Database	org.jboss.security.auth.spi.DatabaseServerLoginModule
DatabaseCertificate	org.jboss.security.auth.spi.DatabaseCertLoginModule
DatabaseUsers	org.jboss.security.auth.spi.DatabaseServerLoginModule
Identity	org.jboss.security.auth.spi.IdentityLoginModule
Ldap	org.jboss.security.auth.spi.LdapLoginModule
LdapExtended	org.jboss.security.auth.spi.LdapExtLoginModule
RoleMapping	org.jboss.security.auth.spi.RoleMappingLoginModule
RunAs	org.jboss.security.auth.spi.RunAsLoginModule
Simple	org.jboss.security.auth.spi.SimpleServerLoginModule
ConfiguredIdentity	org.picketbox.datasource.security.ConfiguredIdentityLoginModule

SecureIdentity	org.picketbox.datasource.security.SecureIdentityLoginModule
PropertiesUsers	org.jboss.security.auth.spi.
SimpleUsers	org.jboss.security.auth.spi.
LdapUsers	org.jboss.security.auth.spi.
Kerberos	com.sun.security.auth.module.Krb5LoginModule
SPNEGOUsers	org.jboss.security.negotiation.spnego.SPNEGOLoginModule
AdvancedLdap	org.jboss.security.negotiation.AdvancedLdapLoginModule
AdvancedADLdap	org.jboss.security.negotiation.AdvancedADLoginModule
AdvancedADLdap	org.jboss.security.auth.spi.UsersRolesLoginModule

**Tabell 2:** Visar de olika klassnamnen med tillhörande kod alternativ för autentiseringsmoduler.

## Autentisering-JASPI

Är ett alternativ till autentisering och det finns många olika Java API:n för detta. Ett XML exempel för detta finns nedan i tabell 3.

XML exempel
<pre>&lt;login-module-stack name="..."&gt;   &lt;login-module code="..." flag="..."&gt;     &lt;module-option name="..." value="..." /&gt;   &lt;/login-module&gt; &lt;/login-module-stack&gt; &lt;auth-module code="..." login-module-stack-ref="..."&gt;   &lt;module-option name="..." value="..." /&gt; &lt;/auth-module&gt;</pre>

**Tabell 3:** XML exempel på inställningar för autentisering-JASPI.

## Tillstånd

XML alternativ för tillstånd finns i tabell 4 nedan.

XML exempel
<pre>&lt;policy-module code="..." flag="..."&gt;   &lt;module-option name="..." value="..." /&gt; &lt;/policy-module&gt;</pre>

**Tabell 4:** XML exempel på för tillstånd.

Nedan finns de olika "code" alternativen för XML-koden för att specificera implementationsklassen för policy modulen som antingen kan vara hela klassen eller en av de listade i tabell 5.

Enstaka moduler	
Kod	Klassnamn
DenyAll	Org.jboss.security.authorization.modules.AllDenyAuthorizationModule

PermitAll	Org.jboss.security.authorization.modules.AllPermitAuthorizationModule
Delegating	Org.jboss.security.authorization.modules.DelegatingAuthorizationModule
Web	Org.jboss.security.authorization.modules.WebAuthorizationModule
JACC	Org.jboss.security.authorization.modules.JACCAuthorizationModule
XACML	Org.jboss.security.authorization.modules.XACMLAuthorizationModule

**Tabell 5:** Visa de enstaka alternativen som finns istället för att inkludera hela klassen.

## Kartläggning

XML exempel för kartläggning finns nedan i tabell 6.

XML exempel
<pre>&lt;mapping-module code="..." type="..."&gt;   &lt;module-option name="..." value="..." /&gt; &lt;/mapping-module&gt;</pre>

**Tabell 6:** XML exempel för kartläggning för "secure-domain".

Kodattributen används för att specificera vilken implementationsklass av inloggningsmodulerna som ska användas eller hela klassen, för att specificera använd de passande av klasserna nedan i tabell 7.

Enstaka moduler	
Kod	Klassnamn
PropertiesRoles	org.jboss.security.mapping.providers.role.PropertiesRolesMappingProvider
SimpleRoles	org.jboss.security.mapping.providers.role.SimpleRolesMappingProvider
DeploymentRoles	org.jboss.security.mapping.providers.role.DeploymentRolesMappingProvider
DatabaseRoles	org.jboss.security.mapping.providers.role.DatabaseRolesMappingProvider
LdapRoles	org.jboss.security.mapping.providers.role.LdapRolesMappingProvider

**Tabell 7:** Visa de enstaka alternativen som finns för kartläggning.

## Granskning

XML exempel av anpassning för granskningsmodulen kan hittas nedan i tabell 8.

XML exempel
<pre>&lt;provider-module code="..."&gt;   &lt;module-option name="..." value="..." /&gt; &lt;/provider-module&gt;</pre>

**Tabell 8:** Visa XML exempel för anpassning för granskningsmodulen.

## JSSE

XML exempel av implementation för JSSE i JBoss EAR filer finns i tabell 9 nedan.

XML exempel
<pre>&lt;jsse keystore-url="..." keystore-password="..." keystore- type="..." keystore-provider="..." keystore-provider- argument="..." key-manager-factory-algorithm="..." key-manager- factory-provider="..." truststore-url="..." truststore- password="..." truststore-type="..." truststore-provider="..." truststore-provider-argument="..." trust-manager-factory- algorithm="..." trust-manager-factory-provider="..." client- alias="..." server-alias="..." service-auth-token="..." client- auth="..." cipher-suites="..." protocols="..."&gt;   &lt;additional-properties&gt;x=y     a=b   &lt;/additional-properties&gt; &lt;/jsse&gt;</pre>

**Tabell 9:** XML exempel av JSSE elementet.

De olika attributen för JSSE är alla friviliga och syns nedan i tabell 10.

Attribut för JSSE	
Attribut	Förklarande text
keystore-password	Lösenord till nyckellagringen.
keystore-type	Vilken typ av nyckellagring som används. Standard är "JKS".
keystore-url	URL till nyckellagrings filen.
keystore-provider	Nyckellagrings leverantör om null JDK leverantören för nyckellagrings typen som används.
Keystore-provider-argument	Sträng som kan användas som argument till leverantörens konstruktor.
key-manager-factory-algorithm	Vilken algoritm som nyckel fabrik hanteraren använder. JDK algoritmen används om den är null.
key-manager-factory-provider	Leverantör av nyckel fabrik hanteraren. Om NULL JDK's nyckel fabrik hanterare.
truststore-password	Lösenord för tillförlitlig lagring.
truststore-type	Vilken typ av tillförlitlig lagring som används, om null JKS.
truststore-url	URL till vart den tillförlitliga lagringen finns.
truststore-provider	Leverantör av tillförlitliglagring, om null JDK's

truststore-provider-argument	Sträng som kan användas som argument till leverantörens konstruktor.
trust-manager-factory-algorithm	Vilken algoritm som tillförlitliga fabrik hanteraren använder. JDK algoritmen används om den är null.
trust-manager-factory-provider	Leverantör av tillförlitliga fabrik hanteraren. Om NULL JDK's nyckel fabrik hanterare.
client-alias	Alias på nyckeln som ska användas när klientsidans SSL socket ska skapas.
server-alias	Alias på nyckeln som ska användas när serversidans SSL socket ska skapas.
service-auth-token	Valideringsbevis för att aktivera tredjepartsservice för att ta emot nyckellagringsnyckeln. Används oftast för att ta emot en privat nyckel för signeringssyfte.
client-auth	Flagga för att indikera om serversidan SSL socket ska kräva klient autentisering. Standard är falskt.
cipher-suites	Kommaseparerad lista med chiffer sviter som ska användas av SSLContext.
protocols	Kommaseparerad lista med SSL protokoll som ska användas av SSLContext.

**Tabell 10:** Visar de olika frivilliga alternativen som finns för JSSE.

# Bilaga H: risker och konsekvensförklaring för vardera hot.

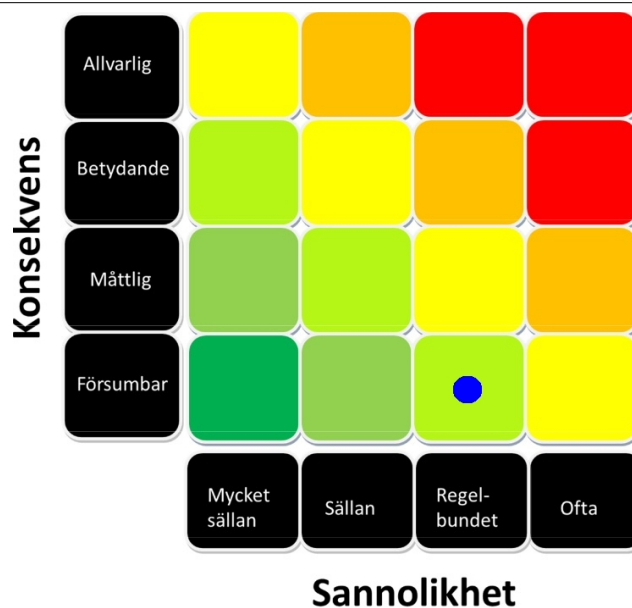
## Överbelastning (t,ex d-dos)

Överbelastnings attacker är de som överbelastar servrar/nätverk.

### Ranking mellan (1-4)

Konsekvens	Sannolikhet
1	3

Konsekvens: förklaring.
Intern: handläggare kan fortfarande arbeta.
Extern: e-tjänster kommer ner.



## Virus

De virus som upptäcks än så länge är inte varit direkt riktade mot Bolagsverket, oftast för att få en ny zombie i ett botnet.

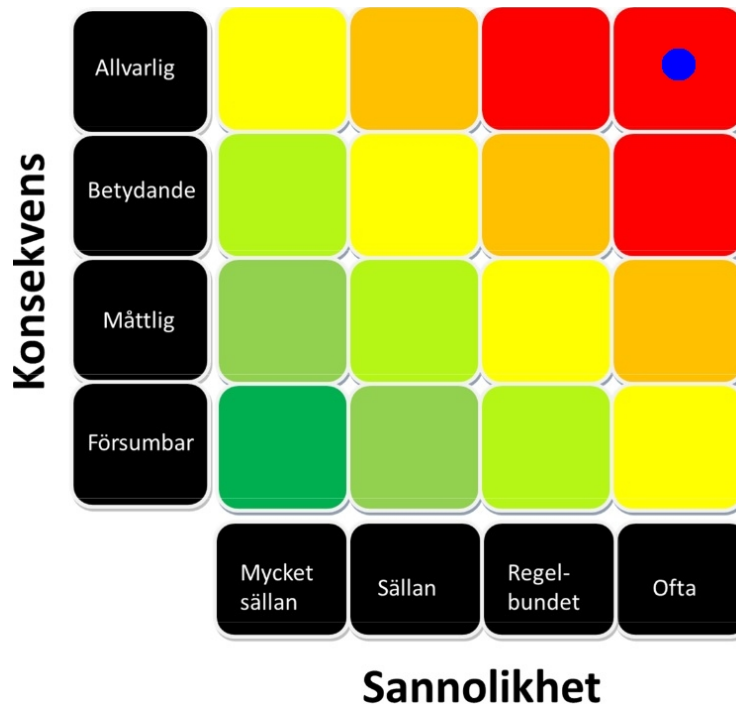
### Ranking mellan (1-4)

Konsekvens	Sannolikhet
4	4

### Konsekvens: förklaring.

Kan vara väldigt stora. Allt ifrån ”remote desktop” till interndator, om virus et får ligga oupptäckt länge kan det i värsta fall manipulera databaser (t.ex. Ändra ägare på bolag).

Men väldigt beroende på virus.



## Wifi

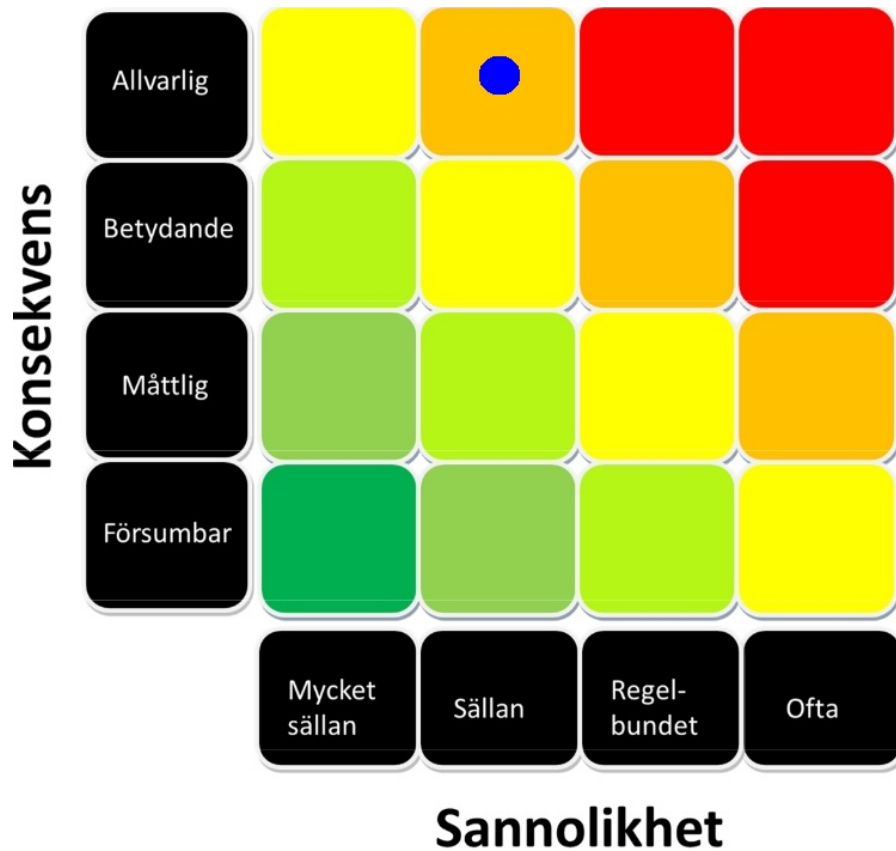
Nu varande skydd är att de använder senaste AC standard.

### Ranking mellan (1-4)

Konsekvens	Sannolikhet
4	2

**Konsekvens: förklaring.**

Idag kommer åt miljöerna. Men i snar framtid kommer vara in boxad av brandvägar.



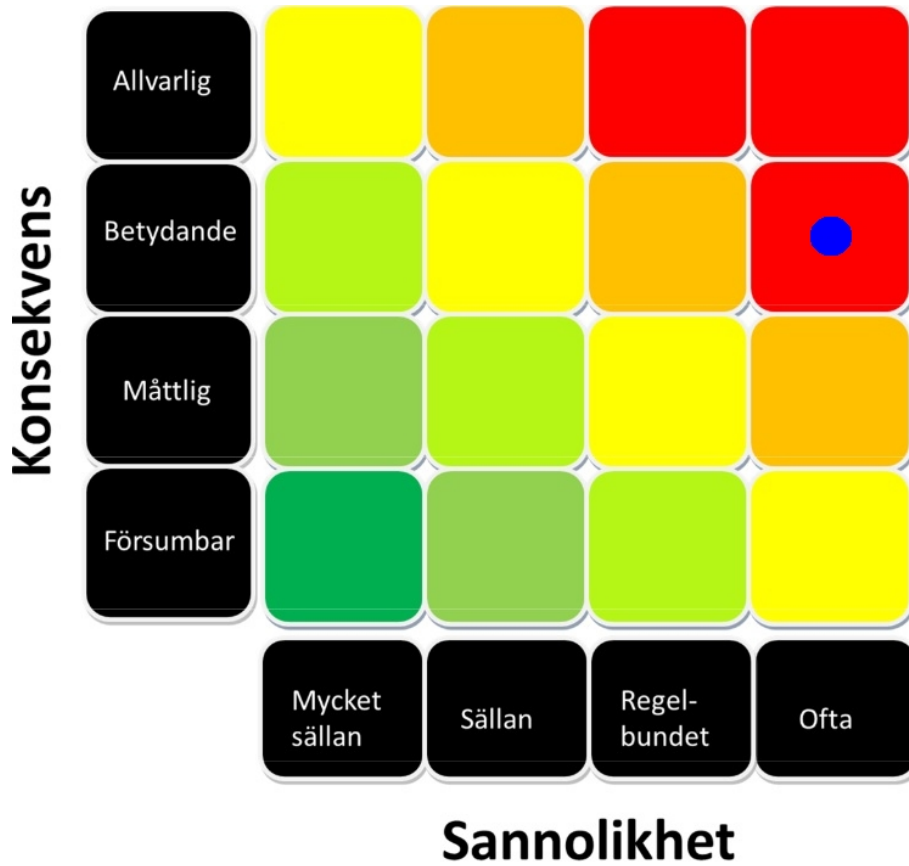
## Webb sidor

### Ranking mellan (1-4)

Konsekvens	Sannolikhet
3	4

**Konsekvens: förklaring.**

När en anställd går ut på en hemsida som innehåller skadlig kod. Få virus och/eller eventuellt lösenord kommer ut.



## Dator stöld

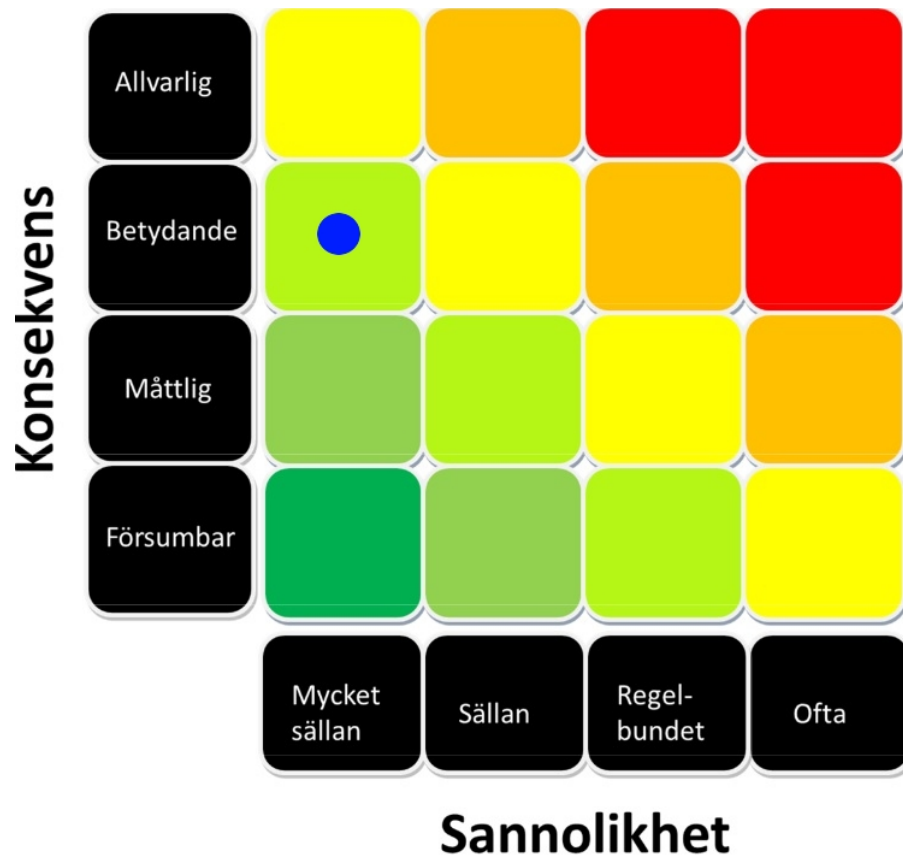
Bärbara datorer är krypterade, snart är även stationära datorer krypterade.

### Ranking mellan (1-4)

Konsekvens	Sannolikhet
3	1

**Konsekvens: förklaring.**

Se programvaror som hjälper de att öppna och förstå vilka filer som används. Då t.ex [H://](#) (extern server lagring) är bara 1 GB så antas att mycket sparas lokalt på datorn. Vilket gör att någon som kommer åt en dator kan då eventuellt komma åt filer som inte ska komma åt. I dag finns ingen kryptering på de stationära datorerna, men laptop:s har krypterad hårddiskar.



## Operativ systems

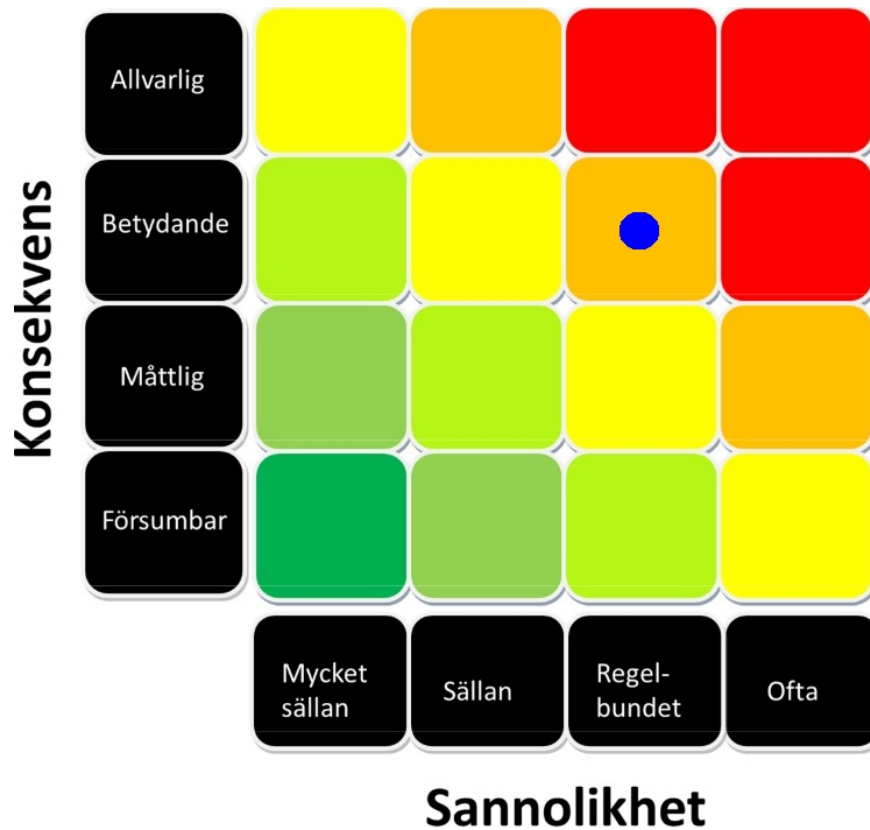
Patchar OS, lättare och snabbare att patcha Windows jämfört med Linux.

### Ranking mellan (1-4)

Konsekvens	Sannolikhet
3	3

**Konsekvens: förklaring.**

Det beror på buggen, men senaste stora buggarna som "one bit to rule them all" eller "shellshock" som dessutom handlar om att vänta på en patch som tar bort buggen.



## Fysik installation av skadlig kod

Install on the go är att någon pluggar in en USB eller annan extern lagring med skadlig kod i en dator.

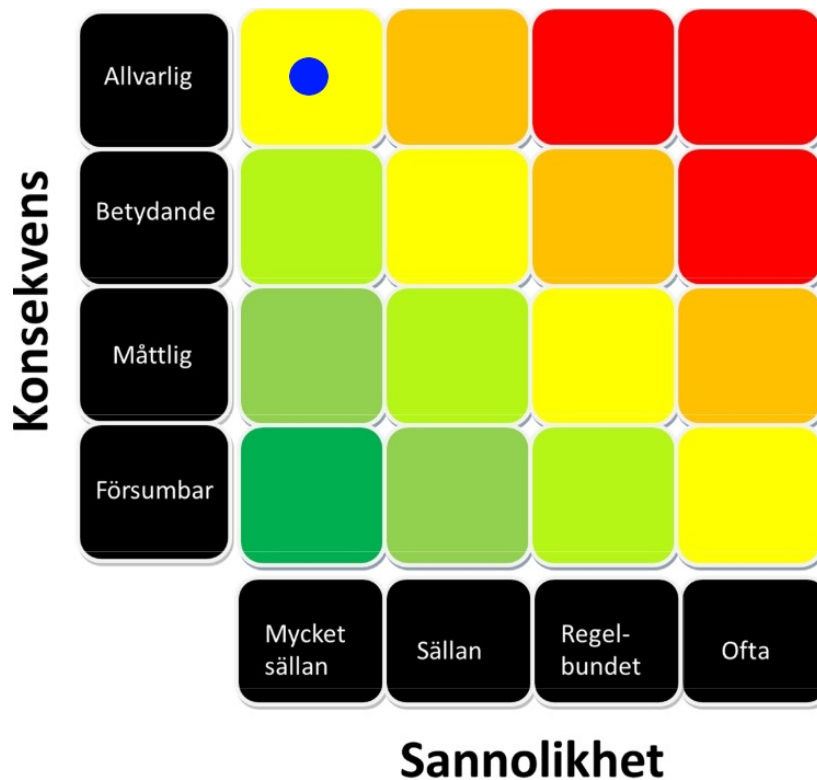
Nu varande skydd realtids anti-virus på exekvering av USB. Låg sannolikhet då person måste få någon annan eller själv plugga in fysiskt USB-minne som har den skadliga koden som anti-virus inte känner till.

### Ranking mellan (1-4)

Konsekvens	Sannolikhet
4	1

**Konsekvens: förklaring.**

Antivirus körs realtid, om inte antiviruset hitta den skadliga koden så är det samma som risken för virus/worms/spyware.



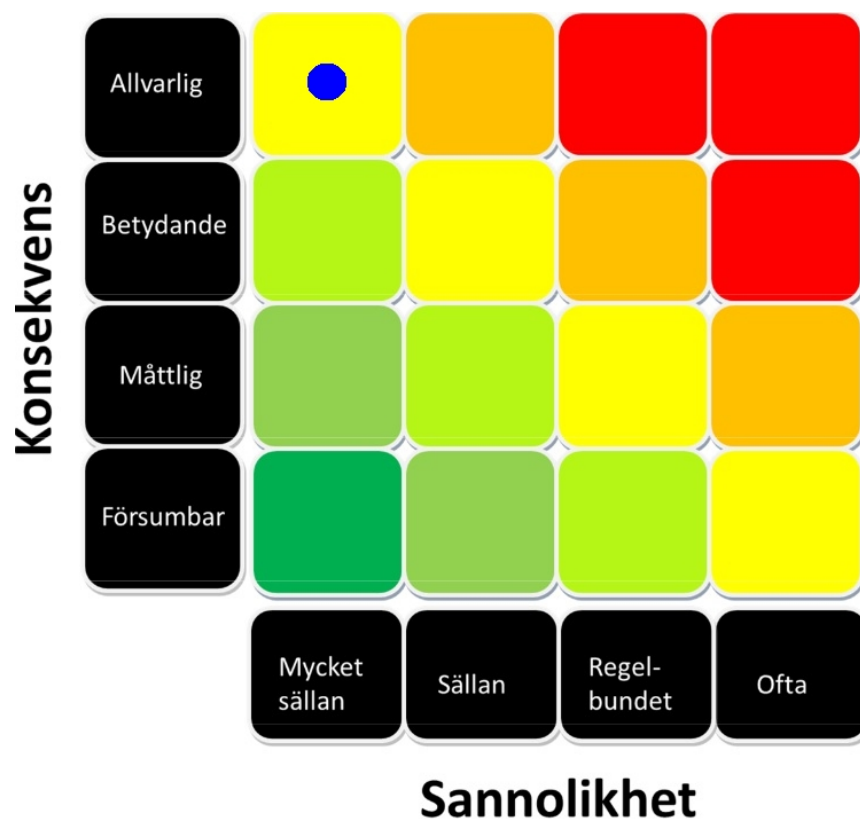
## Registerbevis

Borttagning på grund av sekretess.

### Ranking mellan (1-4)

Konsekvens	Sannolikhet
4	1

Konsekvens: förklaring.
Går att förfalska bevis och kan då låna pengar på det företaget hos banken.



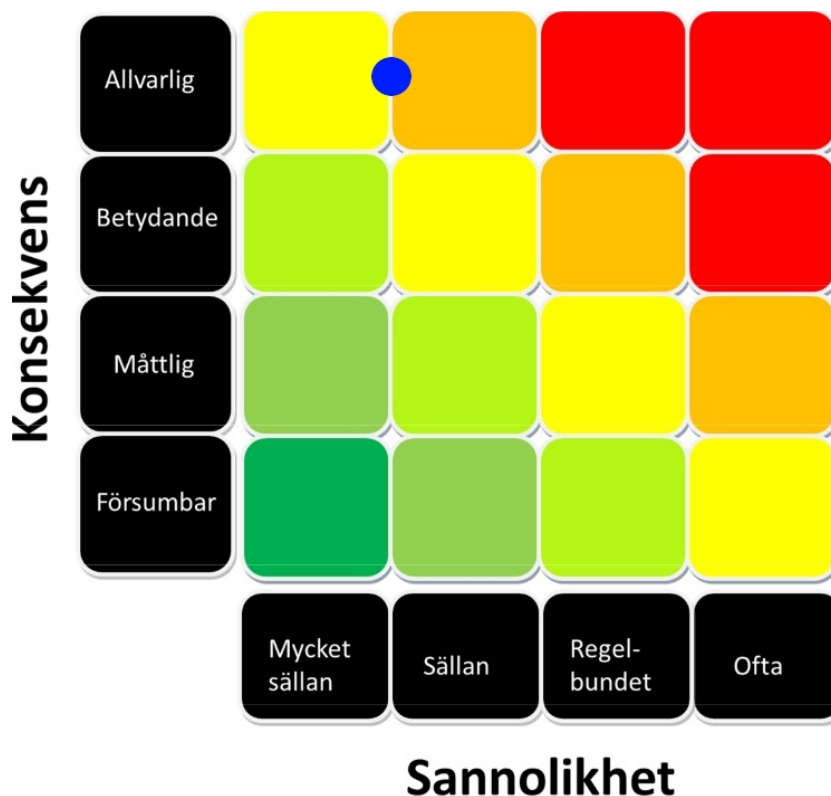
## Säkras koden som används med kryptering

### Ranking mellan (1-4)

Konsekvens	Sannolikhet
4	1.5

### Konsekvens: förklaring.

Då kan en "hacker" i värsta fall läsa koden och skriva dit egna saker som gör något för "hacker" vilket kan skadar Bolagsverket/Organisationer.

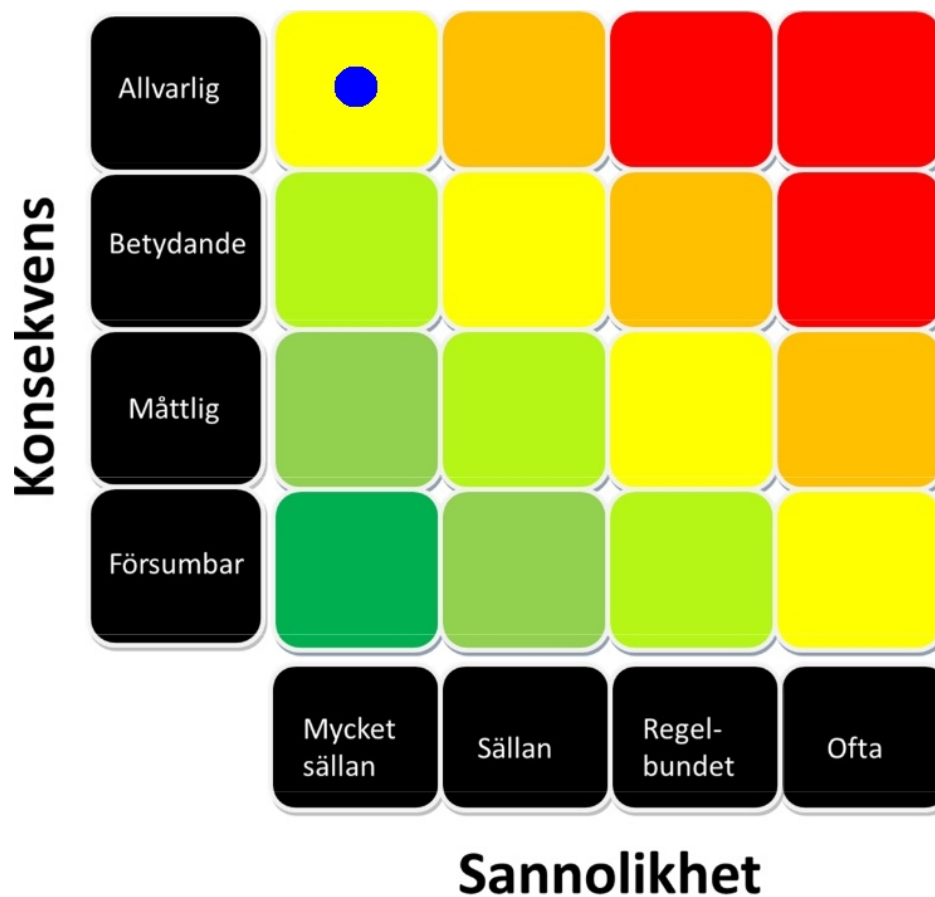


## Lönesystem

Extern tjänst.

### Ranking mellan (1-4)

Konsekvens	Sannolikhet
4	1
Konsekvens: förklaring.	
Borttagning på grund av sekretess.	



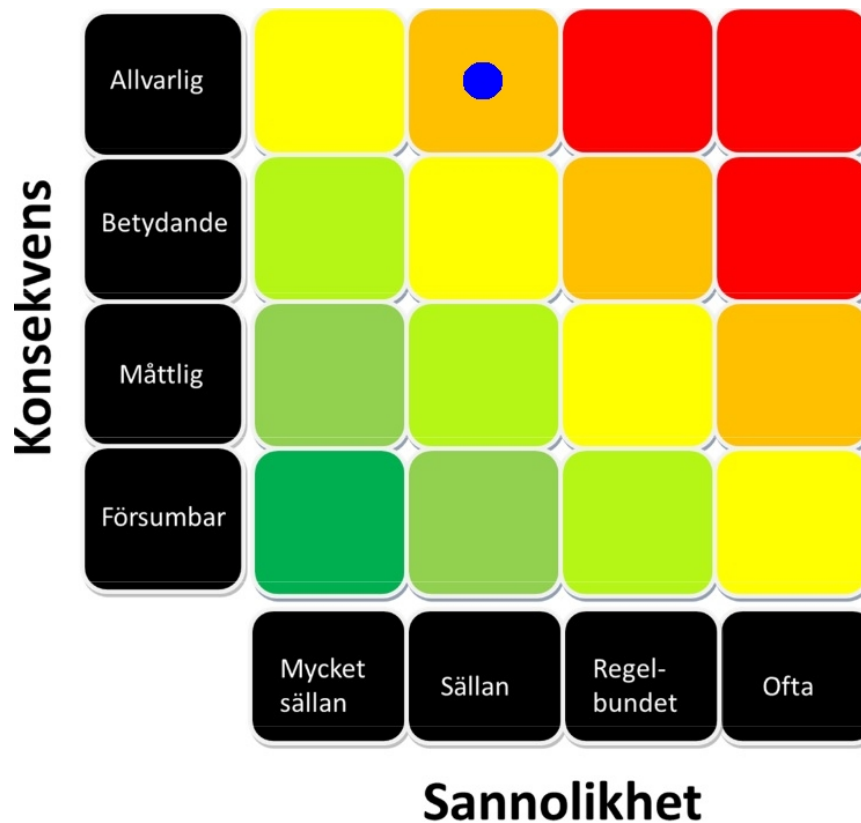
## Nätverks intrång

### Ranking mellan (1-4)

Konsekvens	Sannolikhet
4	2

**Konsekvens: förklaring.**

Om det finns ett nätverks intrång kan de leda till konsekvenser som att ändra databaser och eventuellt förstöra dem.



## Hot och säkerhetsproblem som eventuellt saknas

1. Java versionen går inte att uppgradera (i den takt som önskas), vilket gör att en attack kan nyttja kända buggar i språket.

### Konsekvenserna

1. 3

### Sannolikheten

1. 4

