Institutionen för kommunikation och information
Examensarbete i datalogi med inriktning mot nätverks- och systemadministration 15hp
C-nivå
Vårterminen 2010

Single sign-on

Kerberos i webbapplikationer

Hans Gustafsson Westman

Single sign-on - Kerberos i webbapplikationer

Examensrapport inlämnad av Hans Gustafsson Westman till Högskolan i Skövde, för Kandidatexamen (B.Sc.) vid Institutionen för kommunikation och information. Arbetet har handletts av Jakob Ahlin.

2010-08-19

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

Single sign-on - Kerberos i webbapplikationer

Hans Gustafsson Westman

Sammanfattning

Detta arbete undersöker ett par olika tekniker för att implementera *single sign on* med Kerberos i webbapplikationer. Undersökningen har gjorts på HTTP-autentisering som bygger på Microsofts *NegotiateAuth* och *Cosign* från *University of Michigan*. Dessa två tekniker har undersökts för att se hur de står sig mot varandra på kriterier såsom komplexitet, arbetsinsats och mjukvarukrav.

Resultatet visar att HTTP-autentisering är väldigt simpel att implementera men kräver dock att användarens webbläsare konfigureras för den. Cosign är mer komplext men använder sig av *Cookies* vilket gör att de flesta webbläsare stödjer tekniken utan extra konfiguration.

Nyckelord: Single sign on, Kerberos, NegotiateAuth, HTTP-autentisering, Cosign

Innehållsförteckning

1 Introduktion

Idag finns ett stort utbud av program, funktioner, filer och andra resurser på Internet. En användare idag har ofta användarkonton på flera olika datorsystem för att få tillgång till de tjänster som de behöver och tycker om. Det uppskattas inte av användare att behöva äga och hantera flera användarkonton på flera olika datorsystem. Om användarna får välja själva kommer de att använda samma användarnamn och lösenord på alla datorsystem för att undvika att behöva komma ihåg många olika lösenord. Användarna tycker också att det är besvärligt att behöva autentisera sig en gång för varje tjänst de önskar använda. Det kan exempelvis vara ett nätverk, e-postsystem, bokföringssystem eller flera olika hemsidor. En teknik för att bara behöva logga in en gång i ett datorsystem kallas *Single sign-on* (SSO). Det betyder att en användare autentiserar sig en gång när de först önskar använda en tjänst. Den autentiseringen sparas och skickas vidare till alla andra tjänster som kräver att användaren autentiserar sig (Pfleeger & Pfleeger, 2006).

Enligt Gonsalves (2008), De Clercq (2002), Pashalidis och Mitchell (2003, 2004) finns det ett antal fördelar med att implementera SSO i sitt datorsystem:

- Användare behöver bara komma ihåg en kombination med användarnamn och lösenord. Detta kan också spara tid och arbete för de som arbetar med support för ett datorsystem då de slipper många samtal om förlorade inloggningsuppgifter.
- I vissa datorsystem där samma användarkonto används på flera system kan tid sparas på att bara autentisera användaren en gång.
- Säkerheten stärks genom att användaruppgifter bara skickas en gång vilket minskar möjligheterna för en obehörig part att avlyssna uppgifterna.
- SSO hanteras på en central plats som autentiseringen sker mot. Alla andra tjänster som kräver autentisering skickar en förfrågan mot denna centrala plats först för att kontrollera om användaren redan är autentiserad.

Samtidigt visar Pashalidis och Mitchell (2003, 2004) att det finns ett antal nackdelar med SSO:

- När en användare är autentiserad har den tillgång till alla tjänster den är tillåten att använda. Detta kan vara en sårbarhet om en utomstående part skulle få tillgång till användaruppgifterna.
- Den centrala servern som all autentisering görs mot är en kritisk punkt i datorsystemet. Om den skulle bli otillgänglig kan ingen användare få tillgång till någon av tjänsterna som använder SSO.
- Det kräver att tjänsterna som är inblandade är synkroniserade eller på något sätt samarbetar med varandra och den centrala servern.

Det finns flera tekniker idag för att implementera SSO i olika miljöer. De är dock ofta begränsade för en viss typ av miljö, till exempel klient/servermiljöer eller för webbapplikationer. Tekniker för att hantera SSO i blandade miljöer är något som är ganska dåligt undersökt enligt Mather (2002).

1.1 Syfte

Detta arbete syftar till att undersöka och jämföra olika tekniker för att implementera SSO med Kerberos i webbapplikationer. De ska kunna implementeras i ett datorsystem som innehåller både nätverkstjänster och webbapplikationer. De tekniker

som väljs ska innebära att så få extra komponenter som möjligt ska installeras för att datorsystemet ska vara lätt att administrera och underhålla samtidigt som systemets säkerhetsnivå behålls. Det är också viktigt att teknikerna som väljs fungerar bra för användaren och inte påverkar deras användning negativt. Målet är att användare i systemet bara ska behöva autentisera sig en gång i början av en session för att sen ha direkt åtkomst till alla tjänster den är behörig att använda.

1.2 Relaterade arbeten

Hodges, Howlett, Johansson och Morgan (2008) har skrivit ett arbete där de vill visa hur Kerberos kan passa in i autentiseringen på webben. De har från slutanvändare, administratörer och organisationer tagit fram ett antal kriterier som de anser är viktiga när det gäller säkerhet på webben. Ett antal scenarion skapades av kriterierna för att sätta dem i en kontext. De diskuterar även om ett antal webbrelaterade säkerhetstekniker men de visar också ett antal webbläsare, webbservrar och applikationsmiljöer som stödjer autentisering med Kerberos. Arbetet av Hodges, m.fl. (2008) ger en överblick över utvecklingen av Kerberos på webben och pekar ut vissa problem medan detta arbete ska på ett mer praktiskt sätt undersöka olika tekniker för att implementera det.

Novakov (2006) har skrivit ett arbete där han presenterar några kända implementeringar av SSO för webben. Arbetet ger en överblick om implementeringarnas arkitektur, krav, säkerhet och relevanta funktioner. Novakov (2006) tar upp följande implementeringar:

- *Central Authentication Service* (CAS) är utvecklat av Yale University och är skrivet i Java. Den bygger på en arkitektur som liknar Kerberos fast den använder *cookies* istället för *tickets*.
- WebAuth är utvecklat av Stanford University och är designat att använda en befintlig Kerberos-tjänst.
- CoSign är utvecklat av University of Michigan och innehåller ett helt eget system för autentisering på webben som i sin tur kan kopplas till andra autentiseringsmekanismer, bland annat Kerberos.

Orton (2008) har under en konferens presenterat problem och lösningsförslag för användningen av Kerberos för SSO över *hypertext transfer protocol* (HTTP). Problemen som Orton (2008) tar upp är:

- Hur kan webbservrar integreras i en Kerberos-infrastruktur?
- Hur kan antalet gånger en användare måste ange sina uppgifter minska?

Presentationen är fokuserad på hur Kerberos kan användas på interna nätverk och inte för Internet. Orton (2008) presenterar tre tillgängliga tekniker, *WebAuth*, *HTTP basic authentication* och *HTTP negotiate authentication*, och förklarar övergripande hur de fungerar, vilka krav som ställs, deras fördelar och nackdelar. Han presenterar också några förslag på framtida ändringar för HTTP hur det bättre kan anpassas för autentisering, främst med Kerberos.

2 Bakgrund

Detta kapitel syftar till att ge läsaren en förståelse för de fakta som är relevant för att förstå problemformuleringen. I detta kapitel beskrivs några olika typer och tolkningar av SSO, en kort genomgång av Kerberos samt övriga definitioner som kommer att användas i arbetet.

2.1 Definitioner

Här definieras de termer som är centrala för detta arbete:

Klient: En datorterminal som en person använder och som är uppkopplad till ett datornätverk.

Server: En dator som förser klienter med resurser och tjänster över ett datornätverk.

Session: En tidsperiod som en klient förblir autentiserad, från att autentiseringen skedde tills den avslutas.

Teknik: En teknik i detta arbete innebär det som måste installeras och konfigureras för att SSO ska fungera. Det kan betyda att programvaror och moduler installeras på servern eller klienten. Det kan även innebära att befintlig mjukvara måste konfigureras för att tillåta den nya att fungera.

Webbapplikation: En programvara som en klient kan ansluta till via sin webbläsare över ett nätverk.

Nätverkstjänst: En programvara som exekveras på en server som en klient kan ansluta till över ett nätverk.

2.2 Tolkningar av single sign-on

En vanlig uppfattning om vad SSO innebär är att flera applikationer använder samma lösenord men som lagras separat för varje tjänst. Det betyder att lösenorden kan synkroniseras mellan olika tjänster och system. Ändras ett lösenord skickas det ut till de andra tjänsterna för uppdatering. Denna lösning är smidig för användaren men det är ingen sann SSO. Det finns också en risk med denna typ av teknik. När användaren har samma lösenord på flera olika tjänster räcker det för en obehörig person att få reda på ett lösenord för att ta sig in på alla system användaren har tillgång till.

För sann SSO behöver användaren inte komma ihåg flera lösenord för flera olika tjänster och system. En central server sparar dessa lösenord i en säker databas och de görs tillgängliga när användaren vill autentisera sig. När användaren vill ansluta till en tjänst hämtas uppgifterna från den centrala autentiseringsservern och autentiseringen genomförs i bakgrunden så användaren inte behöver ange användarnamn och lösenord igen. Användaren är då autentiserad på alla tjänster i systemet samtidigt som den ska ha åtkomst till och som stödjer SSO under hela sessionen (Taylor, 2002).

I det här arbetet kommer den senare definitionen att användas.

2.3 Single sign-on på Internet

Taylor (2002) har i sitt arbete om SSO tagit upp de generella begreppen men även övergripande om SSO på webben. Hon skriver bland annat om att mycket på webben idag bygger på portaler som agerar som en central punkt för flera olika webbsidor. När webben började utvecklas var det tillräckligt med *secure sockets layer* (SSL) för att skicka lösenord säkert då säkerheten gick ut på att skydda domänen istället för att

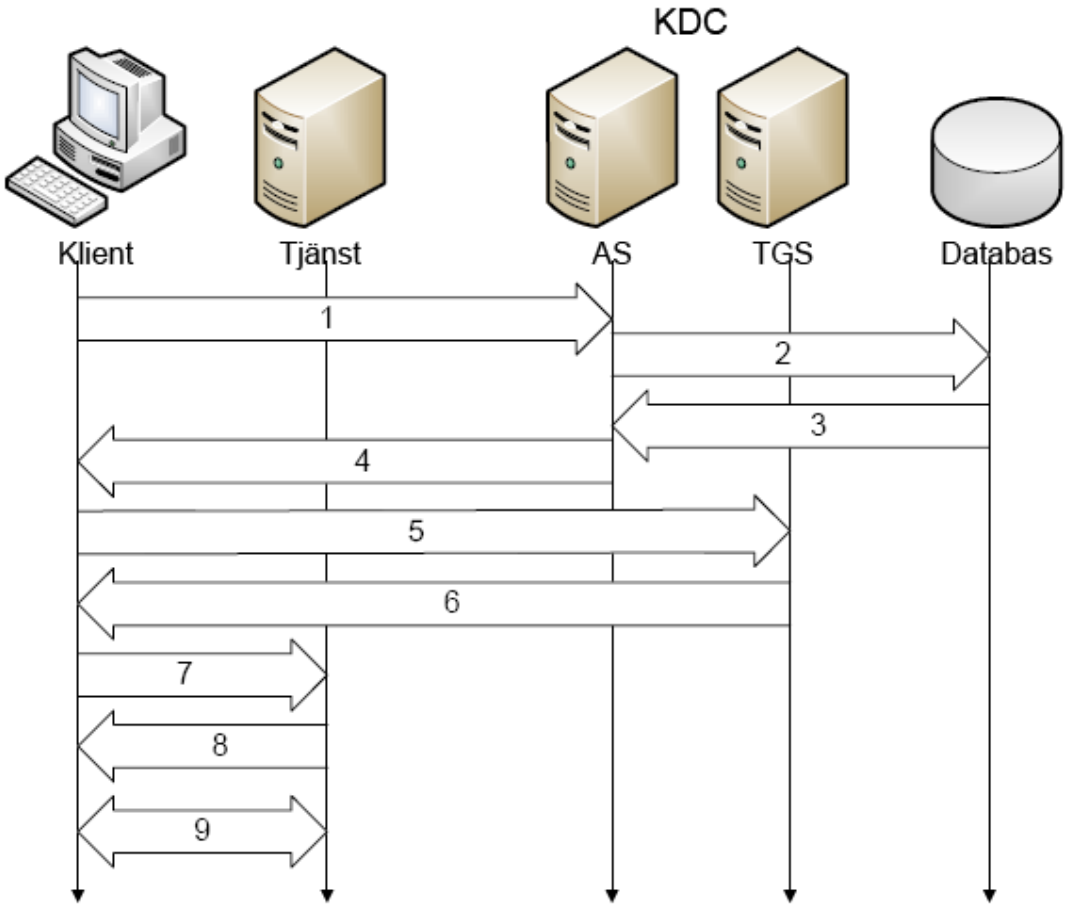
skydda applikationen. När applikationer och databaser började byggas in i webbapplikationerna blev det klart att det fanns begränsningar i SSL. SSL är krävande på en processor och en server kan ofta bara hantera en begränsad mängd SSL-anslutningar. SSL ger själv inte användaren SSO men en portal kan göra att användaren anger sina användaruppgifter en gång och sedan får alla underliggande applikationer den informationen. Autentiseringen sker då i bakgrunden och gäller för alla applikationer som är kopplade till portalen för resten av sessionen. SSL har också begränsningen att det bara ger en säker tvåvägskommunikation så om en transaktion består av tre parter kan inte SSL användas (Taylor, 2002).

Novakov (2006) beskriver i sitt arbete att på stora nätverk finns det ofta webbapplikationer som ska hjälpa användarna och som då också kräver autentisering. I ett sådant fall är det lämpligt och säkert med en centraliserad SSO som är kopplad till det centrala autentiseringssystemet. Ett vanligt exempel på detta är nätverken på universitet och högskolor. SSO på webben bygger då på ett filter som vanligtvis är implementerat på webbservern. Filtrets uppgift är att kontrollera om användaren är behörig att ansluta till den specifika webbapplikationen. Det finns enligt Novakov (2006) två vanliga scenarier för hur autentiseringen sker:

- Användaren får först autentisera sig mot den centrala servern innan den får ansluta till webbapplikationen.
- Användaren försöker först ansluta till webbapplikationen men eftersom den inte är autentiserad skickas den vidare till en tjänst där den kan bli autentiserad.

2.4 Kerberos

Enligt Neuman, Yu, Hartman och Raeburn (2005) är Kerberos är ett protokoll som gör att säker autentisering kan ske på ett osäkert nätverk. Det tillåter både klient och server i ett nätverk att autentisera sig mot varandra. Detta leder till att servern vet att klientens användare är giltig samtidigt som klienten vet att servern är den som den utger sig för att vara. Autentiseringen sker mot central server som kallas *key distribution center* (KDC) som består av två logiska komponenter: en *authentication server* (AS) som sköter autentiseringen och en *ticket granting server* (TGS) som sköter hanteringen av *tickets*. Kerberos använder sig av en *ticket* som klienterna får som ett bevis på en godkänd autentisering. När en användare vill använda en ny tjänst kan dess *ticket* användas som identifikation istället för att utföra en ny autentisering. En KDC har en databas över alla systemets tjänster, användare och deras lösenord.



Figur 1 Autentiseringsfasen i Kerberos

Neuman, m.fl. (2005) och Westman (2010) visar att autentiseringsfasen sker enligt figur 1 och innehåller följande steg:

1. Klienten skapar en egen delad nyckel av användarnamn och lösenord varefter den skickar användarnamnet till AS.
2. AS kollar i en databas om användaren finns.
3. Databasen skickar tillbaka uppgifter om användaren till AS.
4. AS skickar tillbaka en delad nyckel och en *ticket granting ticket* (TGT) till klienten.
5. När användaren vill ansluta till en tjänst skickar den sin TGT till TGS.
6. TGS svarar med *ticket* för den önskade tjänsten.
7. Klienten skickar sin *ticket* till tjänsten.
8. Tjänsten svarar genom att modifiera förfrågningen för att visa att även tjänsten är riktig.
9. Klienten och tjänsten kan nu kommunicera.

2.4.1 MIT Kerberos

En artikel av Calloway (2010) visar att Kerberos skapades av Massachusetts Institute of Technology (MIT) som en del av projektet Athena som startade 1983. Målet var att koppla ihop SSO, nätverksfilsystem, en enhetlig grafisk miljö och en namngivningskonvention. Enligt Danielsson och Westerlund (1998) har USA

restrikterat exporten av MIT Kerberos. Den fick dock exporteras om all krypteringsfunktionalitet togs bort. Denna tomma version av Kerberos har många personer arbetat på för att återskapa funktionaliteten.

2.4.2 Heimdal Kerberos

Heimdal av Danielsson och Westerlund (1998) är en fristående implementering av Kerberos som är fri från restriktionerna som USA sätter på sin export. Den liknar implementeringen av MIT och kompatibel med andra implementeringar. Kerberos version fyra kunde dock exporteras fritt när krypteringsfunktionerna togs bort. Många personer byggde då vidare på den och fyllde igen de bitar som saknades. När Kerberos version fyra började bli för gammal började författarna skriva koden till Heimdal som är en fristående implementering av Kerberos version fem. De har också kollat över en del problem med orginalversionen (Danielsson & Westerlund, 1998).

2.4.3 Microsoft Kerberos

Microsoft har gjort en egen inplementering av Kerberos som Danielsson och Westerlund (2001) jämför mot Heimdal Kerberos under en konferens. Microsoft presenterade sin egen implementering av Kerberos när de lanserade Windows 2000. De gjorde dock små förändringar i den och släppte inte en fullständig dokumentation vilket medförde att det blev svårt att skapa en ersättare till Microsofts Kerberos. Till skillnad från andra implementeringar så lagrar Microsofts Kerberos uppgifterna i en annan databas än vad andra implementeringar använder. Eftersom krypteringen i tidigare versioner av Windows och krypteringen i Kerberos skiljer sig åt implementerade Microsoft en egen kryptering i sin version av Kerberos för att stödja övergången från äldre versioner av Windows till Windows 2000 (Danielsson & Westerlund, 2001).

2.5 Alternativa tekniker för single sign-on

Det finns enligt Gilmore, Farvis och Maddock (2004) några alternativa tekniker för att uppnå SSO utan att använda tekniken som Kerberos använder, till exempel digitala certifikat eller cookies som är en fil som användarens webbläsare laddar ner till klienten. Cookies används ofta på samma sätt som Kerberos och är ett alternativ till *tickets*. Dessa tekniker används främst i samband med SSO mellan webbapplikationer. De flesta webbläsare idag stödjer båda tekniker men det kan också innebära vissa problem. En del användare kan till exempel välja att inte acceptera cookies som webbläsaren försöker spara och då försvinner hela funktionen med SSO.

En annan teknik för att uppnå en stark autentisering är att använda *smart cards*. Det är en fysisk nyckel som har en integrerad krets som kan vara en mikrodator eller liknande med ett internt minne. De är oftast i form av plastkort, nyckelbrickor eller *subscriber identification modules* (SIM). Ett *smart card* används istället för ett lösenord vid autentisering. Autentiseringen sker fortfarande mot en central server fast det går via en kortläsare (Leng, 2009).

3 Problem

Arbetet undersöker och jämför olika tekniker för hur SSO med Kerberos kan implementeras i en webbapplikation som kräver autentisering. Det görs för att ge en uppfattning om hur SSO kan implementeras i en IT-miljö som har både nätverkstjänster och webbapplikationer för att gå mot en mer heltäckande autentiseringslösning. Det görs också för att se i vilka situationer en viss teknik lämpar sig bättre än andra. Arbetet ska ge svar på följande forskningsfråga:

Hur står sig olika tekniker för att implementera SSO med Kerberos i en webbapplikation mot varandra?

3.1 Motivering

Det finns idag flera tekniker för att implementera SSO för nätverkstjänster och det finns flera tekniker för att implementera SSO i bara webbapplikationer. Något som är dåligt utforskat idag är stödet för SSO i en miljö som innehåller både nätverkstjänster och webbapplikationer. Det är något som är eftertraktat idag enligt Mather (2002) och Patnode (2008). Tekniker för att hantera SSO i en blandad miljö har marknadsförts men de har inte levt upp till löftena enligt Mather (2002).

Kerberos är en vanlig teknik för SSO mellan nätverkstjänster men det verkar inte vara välanvänt i samband med webbapplikationer. Istället används alternativ som involverar cookies eller digitala certifikat oftare då de flesta stora webbläsare redan stödjer dem. Arbetet ska undersöka och jämföra tekniker som fungerar i en blandad IT-miljö som använder Kerberos i alla system för SSO. Detta väljs för att uppnå en enhetlig miljö med så få extra tjänster som möjligt. Kerberos väljs för att det är välbeprövat, säkert och det fungerar i alla stora operativsystem såsom Windows, Unix, GNU/Linux och Mac enligt Patnode (2008).

Hodges, m.fl. (2008) visar att det går att implementera SSO med Kerberos i en blandad IT-miljö och de visar att det finns flera programvaror för både klienter och servrar som stödjer detta i en miljö baserad på Kerberos. De anser dock att det är nödvändigt att undersöka en helhetslösning som visar hur de olika komponenterna kan implementeras och fungera tillsammans.

3.2 Avgränsning

Arbetet riktar främst in sig mot hur Kerberos kan fungera tillsammans med en webbapplikation med hjälp av webbaserade autentiseringstekniker. Det riktar alltså inte in sig på hur stöd för Kerberos kan implementeras i andra nätverkstjänster då det ligger utanför arbetets fokus.

3.3 Delfrågor

För att svara på huvudfrågan har den delats upp i två delfrågor. En inledande undersökning måste göras för att ge en bild av vilka tekniker som finns idag för att implementera SSO med Kerberos för webbapplikationer. Det finns som sagt flera tekniker för att utföra det men det krävs en bakgrundsundersökning för att se vilka som kan vara lämpliga att testa i laborationsmiljön. Frågan blir då följande:

Vilka tekniker finns idag för att implementera SSO med Kerberos i webbapplikationer?

De tekniker som tas fram från första delfrågan måste undersökas grundligare för att ge en uppfattning om i vilka situationer de lämpar sig bäst. Det görs också för att se att tekniken fungerar och ger önskat resultat samt att den inte är för komplex att implementera, administrera och underhålla. Frågan som då måste besvaras är:

Hur förhåller sig de olika SSO-teknikerna mot varandra utifrån givna kriterier?

Jämförelsen ska visa hur de valda SSO-teknikerna står sig mot varandra och den är baserad på ett antal givna kriterier som finns specificerade i kapitel 4.2.3.

4 Metod

Metoden för detta arbete består av en förstudie där det kommer att undersökas vilka tekniker som finns och är lämpliga enligt de krav som ställs i kapitel 4.1. En utvärdering ska därefter utföras där de tekniker som väljs kommer att studeras och utvärderas för att se vilka fördelar, nackdelar och användningsområden som finns med varje teknik.

4.1 Förstudie

En litteraturanalys betyder att utvald litteratur analyseras utifrån ett visst mål. I många arbeten kan det vara nödvändigt att göra en djupare studie av litteratur för att till exempel se vilka befintliga lösningar det finns på ett problem idag och sätta dem i kontrast med det egna arbetet (Berndtsson, Hansson, Olsson & Lundell, 2008).

För detta arbete kommer det att undersökas vilka befintliga tekniker som finns och skapa mig en djupare förståelse om dem för att kunna välja ut de som är lämpliga för den IT-miljö som valts. Ambitionen är att hitta tekniker som innehåller så få extra komponenter som möjligt för att datorsystemet ska vara lätt att administrera och underhålla samtidigt som systemets säkerhetsnivå behålls. Det är också viktigt att teknikerna som väljs fungerar bra för användaren och inte påverkar deras användning negativt. Det kommer huvudsakligen att sökas efter information på Internet och i artikeldatabaser.

De tekniker som väljs ska ha stöd för att sköta sin autentisering mot Microsofts Windows Server. Detta är vanligt förekommande i många organisationers datorsystem då de vanligaste operativsystemen för klienter är Windows enligt en undersökning som Netmarketshare (2010) gjort. I Windows Server finns tjänsten Active Directory (AD) med från installation. AD är en katalogtjänst som bygger på *lightweight directory access protocol* (LDAP) och har Microsofts egen implementering av Kerberos inbyggt.

Teknikerna ska också ha stöd för minst en av webbservrarna *Apache HTTP Server* (Apache) och *Internet information services* (IIS). Dessa två webbservrar är idag de största på marknaden enligt Netcraft (2010). IIS följer med Windows Server från början och det är därför en fördel om en teknik stödjer IIS men helst ska de stödja båda webbservrar. Ett krav är också att teknikerna hanterar användarens information på ett sådant sätt att det går att anpassa webbapplikationer till att använda sig av autentiseringsinformationen för egen, intern användarhantering som exempelvis auktorisering och loggning.

4.2 Utvärdering

För att kunna skapa en bas för utvärderingen ska en djupare litteraturstudie utföras för de utvalda teknikerna. Det ska ge den information om varje teknik som krävs för att utvärdera dem utifrån de ställda kriterierna i kapitel 4.2.1. För att få fram konkret information om varje teknik kommer främst respektive tekniks dokumentation att studeras men även de vetenskapliga artiklar som finns tillgängliga kommer att användas.

En alternativ metod är att utföra undersökningen som ett experiment. Berndtsson, m.fl. (2008) har definierat experiment som att undersöka vissa faktorer och se hur dessa påverkas av experimentets kriterier. Vanligtvis används ett experiment för att undersöka om en hypotes är sann eller falsk. Ett experiment kan inte bevisa att en

hypotes stämmer även om resultatet var lyckat. Det beror på att det alltid kan finnas en annan förklaring på problemet. Det finns även mycket som kan gå fel. Resultatet från denna metod kan också utvärderas utifrån kriterierna ställda i kapitel 4.2.1. Om denna metod väljs medför det att teknikerna ska installeras och konfigureras i en laborationsmiljö. Det medför också att en enklare webbapplikation som kräver autentisering måste implementeras för att testa om SSO fungerar. Genom att implementera teknikerna i en laborationsmiljö går det att framställa reell data som visar hur komplext det är att praktiskt installera, konfigurera och underhålla en teknik. Det visar alltså tydligare vilka begränsningar som finns hos teknikerna. Dock kommer ett experiment inte att användas i första hand då den mänskliga faktorn kan spela en stor roll på experimentets genomförande. Ett praktiskt experiment ställer krav om kunskap och erfarenhet på personen som ska utföra det.

4.2.1 Kriterier för utvärdering

Det är ett problem att bedöma en teknik och definiera om den ska bedömas kvantitativt eller kvalitativt. Numeriska värden är oftast lättast att förstå fast i detta arbete är det svårt att utifrån kriterierna bedöma hur väl de uppfylls med ett numeriskt värde. Genom att ha ett antal fördefinierade kriterier är det möjligt att skapa en bra grund med data för varje teknik för att tydligt kunna se hur de olika teknikerna står sig mot varandra. Kriterierna som används är baserade på de som Hodges, m.fl. (2008) har använt då de har ansetts viktiga och relevanta för att utvärdera en SSO-teknik.

Först kommer några kriterier angående teknikernas komplexitet att undersökas. Det görs för att få en bild av hur stor en viss teknik är och på så sätt visa vilken arbetsinsats som kan krävas för att implementera den.

- **Administration**: Hur är den nya tekniken att administrera jämfört med hur datorsystemet fungerade innan SSO implementerades? Om det har blivit mycket svårare att underhålla och administrera datorsystemet efter att SSO implementerats kan det vara svårt att motivera varför det alls ska införas. Det kommer att mäta bedömas utifrån antalet programvaror som kräver löpande administration och underhåll.
- **Extra programvara**: Är det några extra moduler eller programvaror som behövde installeras för att genomföra implementeringen? Om nya komponenter måste installeras i datorsystemet kan det påverka möjligheten att underhålla och administrera systemet samtidigt som det kan leda till fler möjligheter för säkerhetshål. Detta kommer att mätas utifrån hur många extra programvaror eller moduler som måste implementeras för att få SSO att fungera.
- **Funktionalitet**: Fungerer SSO för hela datorsystemet så det räcker med en inloggning eller krävs det fler för att vara fullständigt autentiserad?
- **Autentiseringsmetod**: Teknikernas autentiseringsmetod kommer också tas med i bedömningen för att tydligare visa hur simpla eller komplexa de är men också för att tydliggöra eventuella fördelar eller brister som exempelvis säkerhetshål. Om säkerheten tydligt blivit sämre och det har skapats nya möjligheter för intrång kommer det också bedömas. Säkerheten är dock inget som explicit kommer att undersökas.

Vilka resurser krävs för att implementera SSO? Ambitionen här är att se vilken investering organisationen måste göra för att implementera SSO i sitt datorsystem. Då

en litteraturanalys används som primär metod kommer inte implementationstiden bedömas.

- **Mjukvarukrav**: Vilken mjukvara måste finnas installerad i systemet för att en viss teknik ska kunna implementeras? Detta kommer att bedömas utifrån teknikernas krav på bland annat operativsystem, webbserver och webbläsare.
- **Dokumentation**: Tillgängligheten av dokumentation och eventuella installationsguider kan spela en stor roll för arbetsinsatsen som krävs vid implementering. Det kommer främst att bedömas utifrån tillgängligheten av dokumentation på teknikernas hemsidor.
- **Kostnad**: Har teknikerna någon monetär kostnad? Antagligen kommer ingen teknik innebära extra utgifter för licenser eller programvara men om det finns en kostnad kommer det tas med i bedömningen.

Det kan även uppstå andra oförutsedda fördelar och nackdelar med att implementera en viss teknik jämfört med andra. Det kan vara något som påverkar systemet eller användandet av det. Detta är inget som explicit kommer att mätas men om det uppstår en förändring som tydligt märks av på resultatet kommer det tas med i bedömningen.

5 Resultat

I detta kapitel kommer delmålen från kapitel 4.1 och 4.2 att utföras. Den information som tas fram kommer senare att användas i analysen. Först följer förstudien där två tekniker ska väljas ut som är lämpliga för kravmiljön som finns specificerat i kapitel 4.1. Därefter kommer en utvärdering att göras där teknikerna kommer att studeras mer utförligt för att kunna granskas mot ställda kriterier i kapitel 4.2.

5.1 Förstudie

Först har ett antal vanligt förekommande tekniker tagits fram från utvald litteratur som övergripligt beskrivit flera tekniker. De har sedan granskats djupare om det behövts för att bekräfta att de är lämpliga i kravmiljön. Därefter har två tekniker valts ut som skiljer sig från varandra och som inte verkar vara för komplexa eller för avlägsna från Kerberos.

5.1.1 HTTP-autentisering

Microsoft har skapat ett protokoll för SSO mellan Internet Explorer och IIS som vardagligen kallas *NegotiateAuth* vilket är en av flera implementationer som använder sig av HTTP-autentisering med Kerberos. Detta protokoll utökades senare för att stödja autentisering med Kerberos. Det har på senare tid börjat implementeras i flertalet webbläsare och det finns även som en modul till Apache enligt Jaganathan, Zhu och Brezak (2006) och Wilkinson (2006). *NegotiateAuth* kommer i fortsättningen att refereras till som HTTP-autentisering.

Orton (2008) presenterar en övergripande beskrivning om funktioner, fördelar och nackdelar med HTTP-autentisering. Tekniken skiljer sig från exempelvis Stanfords WebAuth genom att webbservern kan direkt autentisera användare mot KDC utan att behöva gå via en annan tjänst. Det finns enligt Orton (2008) följande fördelar med HTTP-autentisering:

- Det är lätt att installera den och tekniken är inte begränsad av vilken HTTP-klient som används.
- Inga Kerberos-uppgifter skickas över nätverket.
- Det sker en gemensam autentisering där även tjänsterna visar att de är autentiserade.
- SSO blir heltäckande i nätverket. Användarens session gäller på alla servrar och tjänster som stödjer Kerberos.
- Utloggningen sker när Kerberos-sessionen avslutas för att servern har direkt kontakt med KDC.

5.1.2 Cosign

Cosign är ett projekt med öppen källkod från *University of Michigan* vars syfte är att tillåta säker SSO att ske på webbapplikationer. Enligt Xiong (2005) bygger tekniken på ett antal olika moduler varav en är den centrala Cosign-servern som all webbaserad autentisering sker mot. Utloggning görs också genom att besöka en webbplats på den centrala servern. Som ett bevis på en lyckad autentisering använder Cosign *cookies* som sparas i användarens webbläsare. Användaren får följande två *cookies* som innehåller krypterad information om användarens session:

- *Login cookie* som används för den första autentiseringen mot den centrala Cosign-servern i början på en session.
- *Service cookie* som är unik för varje ny webbserver som användaren ansluter till. Den ser till att användaren inte behöver återautentisera sig vid varje ny resursförfrågning.

Denna teknik har då en webbaserad struktur som är väldigt lik Kerberos. Det går enligt Bromberg Craig, Craig och McGowan (2003) och Craig (2006) att koppla Cosign-servern mot en Kerberos-KDC och på så sätt delegera vidare autentiseringsansvaret till Kerberos. Det finns enligt Xiong (2005) följande fördelar med Cosign:

- Cosign tillåter SSO mellan olika domäner.
- Tekniken är inte helt beroende av Cosign-serverns tillgänglighet då den använder *cookies* som finns lagrade i användarens webbläsare. Användaren behöver då inte återautentisera sig så länge som sessionen fortfarande är giltig.
- Alla transaktioner som sker mellan användarens webbläsare, webbservern och Cosign-servern är krypterade med SSL.
- Alla *cookies* som Cosign använder innehåller bara en krypterad slumpad textsträng för att identifiera användarens session så ingen känslig personlig information skickas över nätverket.

Det finns också möjlighet enligt Bromberg Craig, m.fl. (2003) att använda Cosign på både Apache och IIS.

5.1.3 WebAuth

WebAuth är en teknik som är utvecklad av Stanford University och den är uppbyggd i en struktur som liknar Kerberos med en egen webbaserad KDC som kallas *WebKDC*. Tekniken använder sig inte direkt av Kerberos-*tickets* utan använder istället *cookies* på samma sätt som en Kerberos-*ticket*. Dessa *cookies* används då som ett bevis på en lyckad autentisering mellan webbläsare, webbserver och *WebKDC*. Vid autentiseringen anger användaren sina uppgifter i ett formulär i *hypertext markup language* (HTML) som sedan skickas till *WebKDC*. Det är i sin tur *WebKDC* som sköter användarautentiseringen mot Kerberos KDC. Kerberos är dock inte designat för detta syfte då autentiseringen med Kerberos inte når ända ut till klienten. Denna teknik kräver också att trafiken som går mellan webbläsare, webbserver och *WebKDC* är krypterad med SSL för användaruppgifterna ska skickas säkert över nätverket. På grund av att *WebAuth* använder *cookies* blir sessionen begränsad till en webbläsare i taget och för att logga ut innan en *cookies* livslängd gått ut måste de lagrade *cookies* som finns på datorn manuellt tas bort. En annan nackdel med *WebAuth* är att den använder ett HTML-baserat formulär där användarna anger sina uppgifter. Om användarna är vana vid att ange sina uppgifter i ett HTML-formulär kan de bli utsatta för en så kallad *phishing*-attack. Det betyder att en obehörig tredje part skapar ett formulär som ser likadant ut och som ska lura användarna att uppge sina uppgifter (Orton, 2008). Eftersom *WebKDC* använder en modul till Apache som fungerar på ett liknande sätt som modulen HTTP-autentisering använder blir detta bara ett onödigt extra lager i nätverket för att passa in i kravmiljön och därför kommer inte denna teknik att användas.

5.1.4 Shibboleth

Shibboleth är en teknik som på samma sätt som Cosign och WebAuth är en självständig teknik som kan använda sig av Kerberos för autentisering (Franks, 2009). Den består enligt Scavo och Cantor (2005) av de två följande modulerna:

- *Identity provider* (IdP). Den hanterar användarens uppgifter och attribut och tar hand om att autentisera användaren. Den kontrollerar också användarens sessioner.
- *Service provider* (SP). Den skyddar webbresursen från obehöriga användare. Har användaren inte en aktiv session mot *service provider* så skapas en ny session mot *identity provider*.

Fördelen med Shibboleth är att den bara begär och skickar den information om användaren som är nödvändig och tekniken går att använda över flera domäner. Det blir dock samma problem som Cosign har med att Kerberos inte blir heltäckande, sessionen går alltså inte ända ut till klienten. Den måste då också skicka vidare Kerberos-uppgifter vilket Orton (2008) säger att Kerberos inte var designat för.

Shibboleth bygger sin variant av *tickets* på meddelanden i *extensible markup language* (XML) vilket är en öppen standard och har därför bra stöd på de flesta plattformar (Scavo & Cantor, 2005).

Autentiseringen i Shibboleth fungerar på ungefär samma sätt som både Cosign och Webauth. Klienten begär åtkomst till en skyddad webbresurs hos SP. Om användaren redan är autentiserad och har en session igång mot SP får den direkt åtkomst till resursen, annars blir den skickad till IdP. Där kontrolleras det om användaren är autentiserad mot IdP och om den inte är det så utförs en ny autentisering där användaren får ange sina uppgifter i ett HTML-formulär. Om autentiseringen lyckas skickas användaren vidare till SP och får åtkomst till den skyddade resursen (Scavo & Cantor, 2005).

5.1.5 Val av tekniker

Först väljs HTTP-autentisering ut för vidare undersökning då tekniken har stöd av de flesta stora webbläsare och den finns implementerad i en modul till Apache. Den tillåter HTTP-autentisering att ske med Kerberos-uppgifter och är därför lämplig för kravmiljön i kapitel 4.1. Den är också den simplaste teknik av de som undersökts.

Den andra tekniken som väljs ut är Cosign. Det är en större teknik men som samtidigt är mer flexibel och stödjer både Apache och IIS. Det gör att den passar bra i kravmiljön. Tekniken har också gett önskat resultat för *University of Michigan* enligt Bromberg Craig, m.fl. (2003).

De andra teknikerna som undersökts, WebAuth och Shibboleth, har inte valts ut för vidare undersökning. Den främsta anledningen till det är att de är för lika HTTP-autentisering eller Cosign. Det betyder inte att resultatet av undersökningen kan appliceras på dem också men för att resultatet ska skilja sig sig mellan de båda teknikerna har två valts som är ganska olika varandra.

5.2 Utvärdering

I detta kapitel kommer all information om HTTP-autentisering och Cosign att sammanställas enligt uppsatta kriterier. De är två av de tekniker som valdes ut som ett svar på första delfrågan i kapitel 3.3. För att svara på den andra delfrågan kommer

varje tekniks komplexitet, resurskrav, fördelar och nackdelar att beskrivas. Den sammanställda datan för varje teknik kommer sedan jämföras i kapitel 5.3.

5.2.1 HTTP-autentisering

HTTP har inbyggt stöd för autentisering och auktorisering. När autentiseringen önskas svarar webbservern med felkoden 401 som betyder att klienten inte är behörig att ansluta till webbapplikationen. Samtidigt skickar den med instruktioner för hur webbläsaren ska hantera autentiseringen. När klientens webbläsare får felkoden visar den ett dialogfönster för användaren där denna får ange ett användarnamn och lösenord. Klienten försöker då ansluta till webbapplikationen igen fast den skickar med användaruppgifterna i förfrågningen. Webbservern tar emot uppgifterna och fattar ett beslut om den användaren är auktoriserad att använda tjänsten. Det är webbservern som bestämmer vilken autentiseringsmetod som ska användas. Klientens webbläsare anpassar sig efter servern och skickar uppgifterna enligt kriterierna som ställs (Williams & Lane, 2004).

HTTP-autentisering sköter enligt Jaganathan, Zhu och Brezak (2006) och Wilkinson (2006) den första autentiseringsfasen för Kerberos men ser inte till att anslutningen är säker. Den måste istället säkras med SSL. Det kan krävas av vissa webbläsare att de konfigureras för att de ska fungera bra. En del webbläsare har till exempel en lista över webbsidor som är tillåtna att använda SSO. Det krävs också att klienten har stöd för Kerberos och är konfigurerad att använda det.

Jaganathan, Zhu och Brezak (2006) säger att HTTP-autentisering använder sig av *simple and protected GSSAPI negotiation* (SPNEGO) för att förhandla fram en mekanism i *generic security service application program interface* (GSSAPI) för autentisering mellan klient och server. GSSAPI fungerar alltså som ett gränssnitt mellan applikationer och underliggande säkerhetsmekanismer enligt Linn (2000). SPNEGO används av två noders applikationer som använder GSSAPI för att se om de använder gemensamma säkerhetsmekanismer. SPNEGO kan sen välja ut en gemensam mekanism som båda noder kan använda. Det är mest användbart när applikationernas implementering av GSSAPI har stöd för flera säkerhetsmekanismer (Baize & Pinkas, 1998).

Det finns enligt Williams och Lane (2004) följande fördelar med HTTP-autentisering:

- Det är lätt att använda. Att skydda sin webbapplikation med HTTP-autentisering innebär inte mer arbete än att konfigurera webbservern eller skapa en textfil med några få kommandon i.
- Det går att låta en applikation skriven i PHP ta över HTTP-autentiseringen och själv kontrollera användaruppgifterna vilket också fungerar om det är formulärbaserad autentisering.
- Många webbläsare har inbyggt stöd för att samla in och spara användaruppgifterna för HTTP-autentiseringar.
- Det fungerar bra tillsammans med applikationer som är *stateless*. Det betyder att en applikation inte tar hänsyn till tidigare händelser till exempel om användaren nyss autentiserade sig.

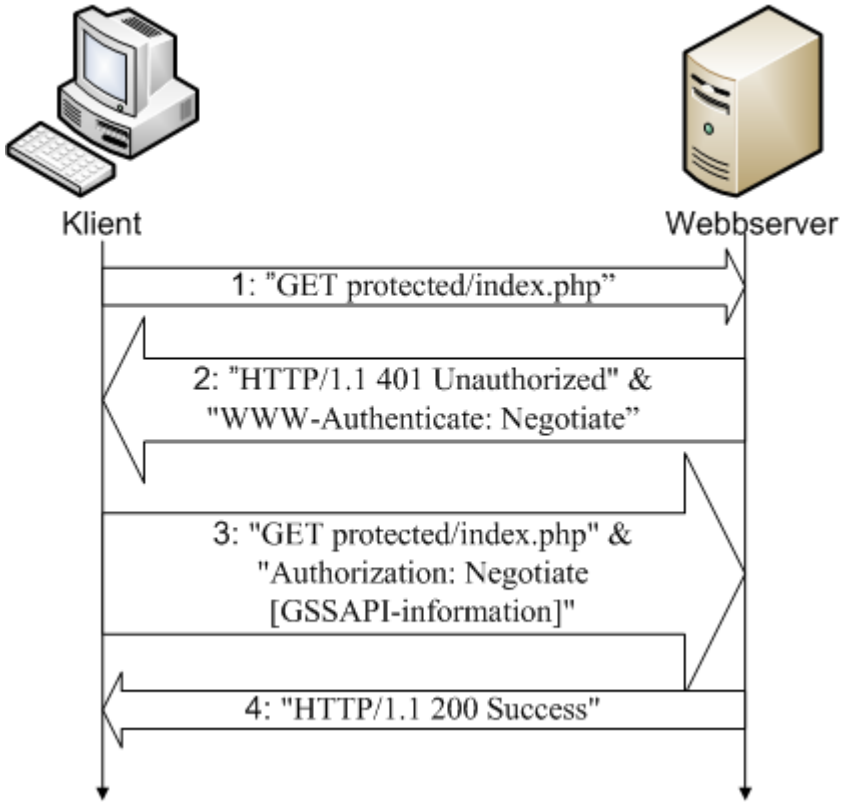
Williams och Lane (2004) säger också att det finns följande nackdelar:

- Det kan vara svårt att använda HTTP-autentisering i applikationer som håller koll på en användares session.

- Webbläsare håller igång en användares session tills nya uppgifter anges eller när användaren loggar ut. En öppen session kan innebära en säkerhetsrisk om användaren lämnar klienten.
- Autentiseringen blir begränsad till webbläsarens dialogfönster vilket begränsar en webbapplikations autentisering till användarnamn och lösenord.
- HTTP-autentisering stödjer bara en auktorisering per autentisering. En webbapplikation kanske kräver flera olika inloggningar för att komma åt mer restrikterat material och HTTP-autentisering har inte stöd för detta.

Jaganathan, Zhu och Brezak (2006) säger att denna sorts autentisering sker med följande steg som också finns illustrerade i figur 3:

1. Användaren skickar en förfrågning om en skyddad resurs till servern.
2. Servern svarar med felkoden ”401 Unauthorized” och skickar med fältet ”WWW-Authenticate:” i HTTP-paketet som visar att servern kräver autentisering. Fältet innehåller ingen inkluderad information.
3. Klienten svarar med att skicka tillbaka samma förfrågning fast i fältet "WWW-Authenticate:" har information från GSSAPI inkluderas.
4. Om allt gått rätt till svarar servern med meddelandet "HTTP/1.1 200 Success" vilket tyder på en lyckad autentisering och auktorisering.



Figur 2 NegotiateAuth

HTTP-autentisering kräver att modulen *mod_auth_kerb* installeras till Apache. För att modulen ska kunna kommunicera med AD och Kerberos behöver den en *keytab*. Det är kort beskrivet en krypterad, lokalt lagrad kopia av webbserverns nyckel hos KDC. Den måste genereras med ett program i AD och sen flyttas över till webbservern. För att koppla modulen till Kerberos måste den konfigureras. Det krävs dock bara några få kommandon i en fil för detta. Den behöver veta vilken Kerberos-domän som ska

användas, var *keytab*-filen finns samt några detaljer om autentiseringen. Installation och konfiguration finns väl dokumenterat av Kouril (2004) och det finns flera installationsguider. Det går att bestämma direkt i konfigurationen vilka kataloger på webbservern som ska använda sig HTTP-autentisering.

En stor fördel med tekniken är att Kerberos blir heltäckande, autentiseringen går från KDC ända ut till klienten för att Kerberos-uppgifterna bara förmedlas vidare via GSSAPI. Modulen i sig kräver ingen vidare administration eller drift om inte nätverksstrukturen förändras. All användarhantering sköts centralt hos AD eller annan katalogtjänst. Kerberos inbyggda säkerhet blir också bevarad då det inte finns någon mellanhand som stör kommunikationen mellan klient, webbserver och KDC. Modulen förmedlar bara vidare uppgifterna. Om användaren inte redan är autentiserad mot KDC så blir den frågad av webbservern om uppgifterna.

Vid en lyckad autentisering skapas ett antal extra servervariabler som bland annat innehåller användarens användarnamn. Servervariablerna är ett antal parametrar som skickas med varje HTTP-förfrågan. De kan hämtas från webbapplikationer som bygger på exempelvis *active server pages* (ASP) eller *hypertext preprocessor* (PHP) med hjälp av inbyggda funktioner för detta syfte. Det finns inget krav för HTTP-autentisering att skydda trafiken med SSL.

Modulen är enbart utvecklad för Apache men då webbservern finns till alla stora operativsystem kan de också använda modulen. Det ställs därför inga direkta krav på vilken hårdvara och operativsystem som måste köras. Om det redan finns en nätverksmiljö med Kerberos och Apache installerat går det snabbt att implementera modulen. Den kräver också bara enklare konfiguration så implementeringen går snabbt och den kommer att kräva minimalt med underhåll. Modulen är också *open source* vilket betyder att den är gratis och får användas fritt.

NegotiateAuth som HTTP-autentisering använder finns definierad i *request for comments* (RFC) nummer 4559 vilket betyder att specifikationerna finns väl dokumenterade och är tillgängliga för vem som helst att ta del av.

Det finns dock en nackdel med tekniken enligt Orton (2008). Om användaren inte redan är autentiserad mot KDC innan de begär åtkomst till en skyddad webbapplikation blir de efterfrågade om sina Kerberos-uppgifter i en dialogruta som webbläsaren presenterar. Användarna kan bli vana med att ange sina uppgifter i en dialogruta och de blir då mer sårbara för en *phishing*-attack.

5.2.2 Cosign

Cosign är enligt Xiong (2005) och Bromberg Craig, Craig och McGowan (2003) uppbyggt primärt av tre moduler: en *CGI*-modul, en *Daemon*-modul och ett filter.

CGI-modulen hanterar främst användargränssnittet men den är även ansvarig för att registrera varje tjänst som användaren loggar in på för att knyta användarens *login cookie* mot användarens session på webbapplikationen.

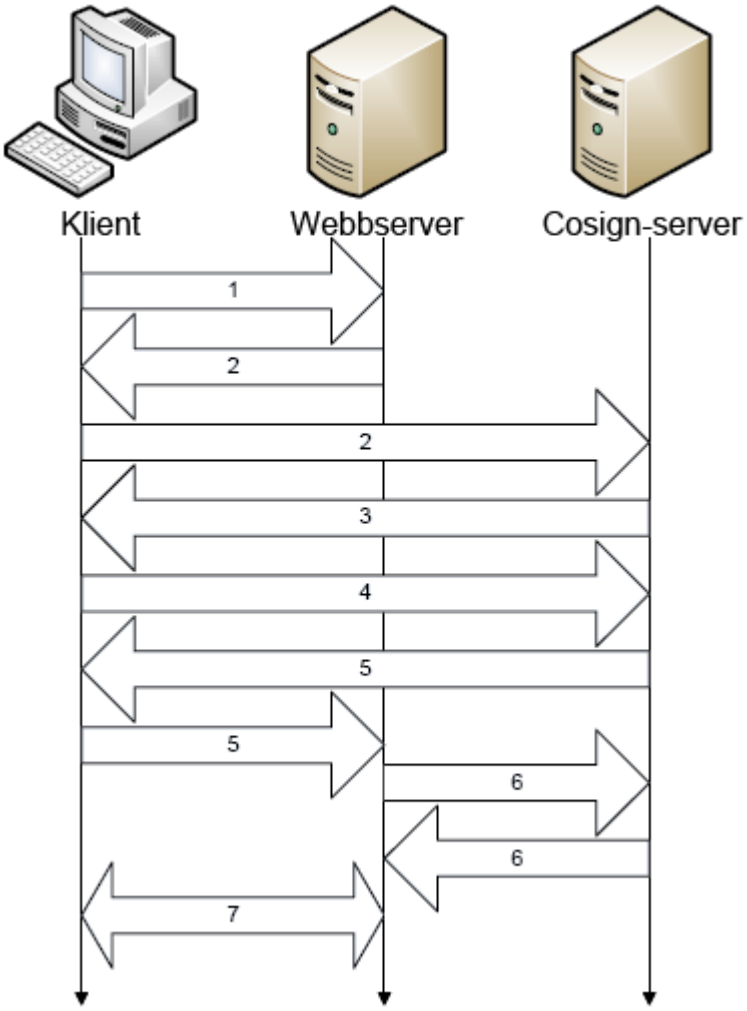
Daemon-modulen är en bakgrundstjänst som körs på Cosign-servern. Modulen är ansvarig för att kontrollera alla användarsessioner och hålla reda på alla *service cookies* som är aktiva. Den svarar också på frågor om en användares identitet från både *CGI*-modulen och filtret och den kan även kommunicera med andra *Daemon*-moduler.

Cosign använder sig också av en typ av filter på webbservern för att kontrollera alla åtkomstförfrågningar till en viss skyddad resurs. Den kontrollerar alltså om

användaren är autentiserad mot Cosign-servern innan den beviljas åtkomst till webbapplikationen. Filtret kan även samla in viss information om användaren som till exempel kan användas vid auktorisering av användaren.

Enligt Xiong (2005) består autentiseringsfasen i Cosign av följande steg som också finns illustrerade i figur 4:

1. En icke autentiserad användare begär åtkomst till en skyddad webbapplikation.
2. Filtret kontrollerar om det redan finns en *service cookie* för tjänsten. Om användaren inte har anslutit till tjänsten tidigare skapar filtret en ny *service cookie* och skickar vidare användaren till Cosign-servern.
3. Cosign-servern kontrollerar om användaren har en giltig *login cookie*. Om den inte finns kan inte användarens *service cookie* registreras. Cosign-servern skapar då en ny *login cookie* och visar ett autentiseringsformulär för användaren.
4. Användaren anger sitt användarnamn och lösenord som skickas tillbaka till Cosign-servern tillsammans med *login cookie* och information om tjänsten.
5. Cosign-servern verifierar användarens uppgifter. Om autentiseringen lyckas så associerar servern *login cookie* med *service cookie* och skickar användaren vidare till webbapplikationen tillsammans med sin *service cookie*.
6. Filtret kontrollerar åter igen användarens *service cookie* mot Cosign-servern. Servern verifierar att användaren är autentiserad och returnerar användarens användarnamn.
7. Filtret granskar svaret, ger användaren åtkomst till den skyddade resursen och sparar lite av användarens information så att den inte ska behöva kontakta Cosign-servern vid varje ny resursförfrågning.



Figur 3 En illustration av autentiseringsfasen för Cosign

För att installera Cosign i IIS krävs först att Cosign-servern, även kallad *Weblogin*, installeras. Den kan vara placerad på en separat maskin om det önskas. Det ställs även krav på att det finns signerade digitala certifikat för att kryptera trafiken mellan klient, webbserver och Cosign-server. Cosign-servern måste konfigureras med URL till tjänsten och webbsidan där användare autentiserar sig. Även kataloger där alla *cookies* och Kerberos-*tickets* sparas måste konfigureras. Sedan kopieras modulen till IIS och installeras.

Filtret installeras på webbservern som tillhandahåller de skyddade resurserna. Det installeras genom att konfigurera en fil med uppgifterna för nätverkets Cosign-miljö och digitala certifikat. Filtret aktiveras genom att appliceras på en webbplats i IIS antingen genom det grafiska verktyget eller med kommandon i kommandotolken. Det finns noggranna installationsanvisningar på projektets hemsida (Cosign, 2009).

Cosign, liksom HTTP-autentisering, kan använda servervariabler för att ge en webbapplikation information om användaren och om den är autentiserad. Cosign lägger även till några egna servervariabler som identifierar autentiseringsinformation, vilken tjänst som använts och eventuellt en sökväg till användarens Kerberos-uppgifter. Cosign är en teknik som fristående sköter autentisering. Cosign-servern kan också förmedla vidare autentiseringen till bland en utomstående tjänst, till exempel AD.

Cosign kan installeras både till Apache och IIS. Även om Apache finns till alla stora operativsystem så är det en fördel att tekniken också finns till IIS då den webbservern finns med i Windows Server. Cosign kräver att två moduler installeras och konfigureras vilket kan ta tid. Det finns dock detaljerade installationsanvisningar för både Apache och IIS vilket kan spara mycket tid. Det finns också krav på att signerade digitala certifikat finns tillgängliga (Cosign, 2009). Cosign är också *open source* vilket betyder att den är gratis och får användas fritt enligt Moody (2001).

Cosign kan enligt Cosign (2009) använda mer än bara Kerberos till autentisering, till exempel digitala certifikat. Tekniken stödjer också flerfaktorautentisering vilket betyder att fler uppgifter än bara användarnamn och lösenord kan användas för att autentisera en användare. Till exempel kan användarnamn och lösenord användas tillsammans med ett digitalt certifikat för att ytterligare styrka en användares identitet.

Cosign har dock nackdelen att den kräver att användaren autentiserar sig med sina Kerberos-uppgifter manuellt första gången den ansluter till en skyddad webbresurs. Det bryter egentligen mot fullständig SSO men användaren är då automatiskt autentiserad mot alla webbresurser den är behörig att använda (Wilkinson, 2006).

5.3 Jämförelse

I detta kapitel kommer den data som togs fram för HTTP-autentisering och Cosign att jämföras mot kriterierna ställda i kapitel 4.2.1. Teknikerna ska jämföras på administration, extra programvara, funktionalitet, autentiseringsmetod, mjukvarukrav, dokumentation och kostnad. Det görs för att se i vilka situationer en viss teknik är lämplig och hur de står sig mot varandra.

HTTP-autentisering är enklare att installera än Cosign då tekniken endast består av en modul som behöver installeras på webbservern. Den kräver också bara en enklare konfiguration för att ställa in vilken Kerberos-domän som ska användas och några uppgifter omkring det. Cosign består av två moduler som först måste konfigureras med uppgifterna för Kerberos-domänen innan de installeras i webbservern.

HTTP-autentisering har en tillräcklig dokumentation och installationsanvisningar skrivna av Kouril (2004) som också är den som är ansvarig för modulen för tillfället. Det finns även andra hemsidor som beskriver och ger installationsanvisningar för modulen. Cosign har noggranna installationsanvisningar som följer med vid nedladdning av modulerna. Cosign verkar dock ha färre installationsguider är HTTP-autentisering, kanske för att det är en större och mer komplex teknik. Metoden som HTTP-autentisering använder finns också specificerat i RFC 4559 av Jaganathan, Zhu och Brezak (2006).

Ingen av teknikerna ska kräva något pågående underhåll. Om något måste konfigureras om blir det lite mer arbete med Cosign då det har mer konfiguration och att modulen måste tas bort från webbservern, konfigureras om och sedan installeras igen. Ska HTTP-autentisering konfigureras om kan det direkt göras i konfigurationsfilen. Det räcker sedan med att webbservern startas om för att den ska läsa in den nya konfigurationen. Cosign är dock inte helt beroende av KDC. Eftersom tekniken fristående kan sköta användarsessioner och använder *cookies* som hanteras av Cosign-servern, webbservern och webbläsaren kan användare som redan är autentiserade mot en webbapplikation fortfarande använda den.

HTTP-autentisering sköter direkt all autentisering mot AD. Det beror på att den kort sagt bara förmedlar vidare Kerberos-uppgifter och *tickets* mellan AD och användaren. Cosign kan själv sköta autentiseringen men den kan även förmedla vidare den biten

till exempelvis KDC i AD. Cosign använder *cookies* för all webbaserad SSO. Det krävs dock att användaren först autentiserar sig mot den centrala Cosign-servern med exempelvis sina Kerberos-uppgifter för att sedan kunna ansluta mot andra webbresurser enligt Wilkinson (2006). Cosign-servern kontrollerar själv att användarens uppgifter stämmer mot en KDC och ger sen användaren *ticket*-liknande *cookies*.

Eftersom båda tekniker ger användarens uppgifter tillgängliga som servervariabler vid en lyckad autentisering kan webbapplikationerna anpassas på samma sätt oavsett vilken av teknikerna som används. Webbapplikationerna måste då anpassas för att använda servervariablerna för att autentisera och auktorisera användaren vilket finns övergripligt beskrivet i bilaga A. Det kan dock vara svårt att anpassa en webbapplikation som själv hanterar autentisering och användarsessioner. Om den första autentiseringen sker mot webbservern blir det begränsat av webbläsarens dialogfönster eller ett HTML-formulär. Enligt Williams och Lane (2004) är webbläsarens dialogfönster begränsat till att bara ta emot användarnamn och lösenord. Om ett HTML-formulär används så öppnar det upp för mer kreativa autentiseringsmöjligheter men då bör SSL användas för att skydda transaktionerna mellan webbläsare och webbserver enligt Thomas (2000).

Modulen som HTTP-autentisering använder finns bara tillgänglig till Apache, men då webbservern finns till alla stora operativsystem orsakar det därför inga större problem (Apache, 2010; Netmarketshare, 2010). Cosign finns både till Apache och IIS vilket är en fördel då IIS finns med i Windows Server och nätverksmiljön kan då hållas homogen. Att nätverksmiljön är homogen betyder att alla produkter kommer från samma tillverkare vilket kan underlätta administration.

Det ställs heller inga krav på speciell hårdvara så länge som Windows Server kan köras på maskinen. Båda tekniker blir också en engångsuppgift att installera och konfigurera så länge inte något ska ändras om i nätverksmiljön. Det blir därför inget löpande underhåll då all autentisering sker centralt på AD. Cosign kräver dock signerade digitala certifikat för att kryptera nätverkstrafiken vid formulärbaserad autentisering.

Då Cosign använder *cookies* mellan webbservrar och slutanvändare fungerar tekniken med alla webbläsare som stödjer dem. Det kan i vissa fall krävas att webbläsaren konfigureras för att tillåta *cookies* då det ibland är blockerat. Enligt Hodges, m.fl. (2008) har de flesta stora webbläsare inbyggt stöd för HTTP-autentisering. Funktionen brukar dock ofta vara låst och kräver att det manuellt låses upp eller att webbplatsen läggs till i en lista med tillåtna interna webbplatser som är tillåtna att använda HTTP-autentisering. En beskrivning om hur detta kan utföras finns i bilaga B.

Eftersom Cosign är större och innefattar fler moduler och mer konfiguration kommer installationen av tekniken ta längre tid än HTTP-autentisering. Det finns dock noggranna installationsanvisningar för Cosign i IIS vilken underlättar installationen avsevärt.

Båda tekniker är *open source* och är därför gratis och får användas fritt. Att ett program är *open source* betyder att programmet och dess källkod är fritt tillgängliga för vem som helst att använda, modifiera och sprida vidare enligt Moody (2001).

Modulen i HTTP-autentisering är begränsad till enbart Kerberos vilket är precis vad uppgiften kräver. Den sköter autentiseringen mot Kerberos bra och arbetar i bakgrunden vilket gör att en användare inte kommer märka av det. Cosign använder

speciella webbsidor som inloggning och utloggning görs mot. Cosign klarar flera typer av autentisering och är mer komplext än HTTP-autentisering. Tekniken är dock mer flexibel för vilken mjukvara och autentiseringsmetod som används.

En fördel med Cosign är att den kan konfigureras till att använda mer än bara användarnamn och lösenord vid autentisering. Den kan alltså involvera fler faktorer som exempelvis användarnamn, lösenord och ett digitalt certifikat. HTTP-autentisering är begränsat till Kerberos-*tickets* eller användarnamn och lösenord via webbläsarens dialogruta. Att Cosign använder HTML-formulär för att autentisera en användare är också dess svaghet. Cosign kräver att användaren först autentiserar sig mot Cosign-servern oavsett om den redan är autentiserad mot KDC. Efter det fungerar SSO över alla webbresurser som stödjer det. Det krävs då alltså två autentiseringar för att användaren ska ha automatisk tillgång till alla resurser på nätverket om Cosign används enligt Wilkinson (2006). Det är då bara HTTP-autentisering som ger full SSO genom hela nätverket.

6 Slutsats

I detta kapitel följer resultatet av jämförelsen som utfördes i kapitel 5.3. Här sammanställs båda tekniker för att visa vilka fördelar och nackdelar de har samt i vilken sorts nätverksmiljö de lämpar sig bäst. Detta görs för att svara på delfrågorna från kapitel 3.3.

Vilka tekniker finns idag för att implementera SSO med Kerberos i webbapplikationer?

Efter litteraturanalysen har det visats sig att det finns flera olika tekniker för SSO i webbapplikationer. Utöver HTTP-autentisering och Cosign som analyserats djupare i arbete finns WebAuth och Shibboleth som på många sätt liknar de första teknikerna. Det finns även flera andra som inte tagits upp i detta arbete som exempelvis *SideCar* från Cornell University, *PubCookie* från University of Washington och *Central Authentication System* (CAS) från Yale University. Ett kommersiellt exempel är Microsofts *Passport Service*.

Hur förhåller sig de olika SSO-teknikerna mot varandra utifrån givna kriterier?

HTTP-autentisering är litet och simpelt. Tekniken går snabbt att installera och konfigurera då det är en modul till Apache som bara kräver enklare konfiguration. Den kopplas enkelt till en KDC och kan fortsätta på en användares redan befintliga Kerberos-session. Den kan därför bidra till att ge full SSO genom hela nätverket, även i webbapplikationer. Den är dock lite begränsad då den antagligen är bäst lämpad i en mindre organisations kontrollerade nätverksmiljö för att den kräver att webbläsarna konfigureras för att använda SSO på detta sätt.

Cosign är mer komplext än HTTP-autentisering men då det finns installationsanvisningar för både IIS och Apache kan tekniken fortfarande vara smidig att installera och konfigurera. Det blir samtidigt mer flexibel eftersom den stödjer både Apache och IIS. Den innehåller en extra modul som också kan kräva konfiguration. En nackdel med Cosign är en användares befintliga Kerberos-session inte kan användas första gången användaren ansluter till en webbapplikation. En inloggning måste först ske på den centrala Cosign-servern med Kerberos-uppgifterna varefter SSO fungerar mellan alla webbapplikationer som är anslutna. Det krävs därför två inloggningar för att kunna vara autentiserad över hela nätverket. Cosign använder *cookies* vilket är fördelaktigt då nästan alla webbläsare idag stödjer dem. Denna teknik blir då mer lämpad i en miljö som främst innehåller webbapplikationer som sköter sin autentisering mot en central server kopplad till AD. Resultaten för båda tekniker finns sammanfattat i tabell 1.

Resultaten som tagits fram för Cosign styrker de resultat som Novakov (2006) har tagit fram. Detta arbete bekräftar också de slutsatser om HTTP-autentisering som Orton (2008) kommit fram till. Detta arbete ger dock extra kompletterande information som kan vara nyttig vid val av en SSO-teknik för webben. Det syns tydligt i denna jämförelse i vilka miljöer de båda teknikerna lämpar sig. Det är dock lite svagare resultat där teknikernas svårighetsnivå på installation mäts eftersom informationen är tagen från litteratur och inte ett praktiskt experiment.

Detta arbete kan användas som stöd vid val av vilken SSO-teknik som ska väljas för en viss typ av nätverksmiljö. Dock har bara två olika tekniker undersökts vilket betyder att detta arbete inte ger en fullständig bild av hur teknikutbudet ser ut. Det finns flera tekniker som både liknar och skiljer sig en del från dessa två tekniker som

tagits upp i arbetet. Det är väl främst de som planerar, installerar, underhåller och administrerar nätverk som kan tänkas ha användning av detta arbete. Det är antagligen mer praktisk nytta av resultatet än teoretisk.

Tabell 1. Sammanfattning av resultatet

	HTTP-autentisering	**Cosign**
Administration	Ingen	Ingen
Extra programvara	En modul som behöver konfigureras	Två moduler som båda behöver konfigureras
Funktionalitet	Fullständig SSO	Det krävs en extra inloggning för fullständig SSO
Autentiseringsmetod	Modulen vidarebefordrar Kerberos-uppgifter	Använder *cookies* i en Kerberosliknande webbmiljö
Mjukvarukrav	Apache	Stödjer både Apache och IIS
Dokumentation	Det finns kortare dokumentation och flera guider	Utförlig dokumentation och installationsanvisningar finns
Kostnad	Gratis	Gratis

7 Reflektioner

I detta kapitel finns personliga åsikter om hur arbetet är, styrkor och svagheter. Det beskrivs också hur arbetet utförts och vad det har fått för konsekvenser.

Arbetet har följt den primära metoden litteraturanalys. Utifrån granskad litteratur har data tagits fram som har varit relevant för att jämföra tekniken mot de ställda kriterierna. Samma sorts data skulle också kunna tas fram med ett praktiskt experiment där varje utvald teknik implementeras i en laborationsmiljö vilket troligtvis skulle ge noggrannare och mer verklighetstrogen data. Litteraturanalysen gav tillräckligt med data för att dra slutsatser och ge svar på problemfrågan men med mer praktisk data kunde arbetet kanske också givit vissa rekommendationer inför framtida implementering. I ett tidigare skede av arbetet var den primära metoden ett praktiskt experiment. Det uppstod dock problem vid installationen då det antagligen var dåligt förberett. Det är därför det istället utförts en litteraturanalys för att ge liknande resultat som förväntats av ett experiment.

Arbetet visar ganska tydligt i vilka situationer HTTP-autentisering och Cosign lämpar sig. Detta kunde dock styrkas ytterligare genom att praktiskt implementera dem i både labborationsmiljö och i nätverksmiljöerna där vardera teknik lämpar sig. Teknikerna kunde också ha utvärderats med en grupp försökspersoner som testar och använder alla resurser i systemet för att se att inloggning, utloggning, SSO och eventuellt att auktorisering fungerar som det ska. Om det funnits mer tid hade minst en extra teknik testats och då gärna en som skiljer sig från de två som undersökts i detta arbete. Det hade också gjorts en lite djupare analys av vardera teknik.

8 Framtida arbete

Som tidigare sagts så skulle en liknande studie kunna göras på fler tekniker i en mer blandad nätverksmiljö. Undersökningen kan gärna vara mer praktiskt inriktat för att ge mer verklighetstrogen data och då visa hur det verkligen är att implementera olika SSO-tekniker.

Då HTTP-autentisering är väldigt simpel och många andra ganska stora och komplexa då de har mycket mer funktionalitet kan det vara intressant att undersöka om det finns bra kompromisser. En stor och komplex teknik kan kanske vara onödigt komplex och funktionsrik för vissa ändamål samtidigt som en liten och simpel teknik kan vara för begränsad.

Referenser

Apache (2010) *archive.apache.org*. Apache.org. Tillgänglig på Internet: http://archive.apache.org/dist/httpd/binaries/ [Hämtad 2010-07-28].

Baize, E. & Pinkas, D. (1998) *The simple and protected GSS-API negotiation mechanism*. Internet Engineering Task Force. Tillgänglig på Internet: http://www.ietf.org/rfc/rfc2478.txt [Hämtad 2010-07-28].

Berndtsson, M., Hansson, J., Olsson, B. & Lundell, B. (2008) *Thesis projects: a guide for students in computer science and information systems* (2:a upplagan). London: Springer Verlag.

Bromberg Craig, J., Craig, W. & McGowan, K. (2003) The quest for web single sign-on at the University of Michigan. Presenterat vid *EDUCAUSE 2003*, Anaheim, California, USA 4-7 November, 2003.

Calloway, D. (2010) *The history of kerberos authentication*. TheWorldJournal. Tillgänglig på Internet: http://www.theworldjournal.com/special/nettech/news/kerberos.htm [Hämtad 2010-04-04].

Cosign (2009) *Cosign: web single sign-on*. Cosign. Tillgänglig på Internet: http://cosign.sourceforge.net/ [Hämtad 2010-07-06].

Craig, W. (2006) Using Kerberos for web authentication. Presenterat vid *AFS & Kerberos best practice workshop*, Ann Arbor, Michigan, USA 12-16 Juni, 2006.

Danielsson, J. & Westerlund, A. (1998) Heimdal - an independent implementation of kerberos 5. Presenterat vid *USENIX 1998 Annual Technical Cenference*, New Orleans, Louisiana, USA 15-19 Juni, 1998.

Danielsson, J. & Westerlund, A. (2001) Heimdal and windows 2000 kerberos - how to get them to play together. Presenterat vid *USENIX Annual Technical Conference*, Boston, Massachusetts, USA 25-30 Juni, 2001.

De Clercq, J. (2002) Single sign-on architectures. I: Y. Frankel, G. I. Davida, & O. Rees (red:er), *Infrastructure Security* (s. 40-58). Heidelberg, Tyskland: Springer-Verlag.

Franks, C. (2009) *Using Kerberos tickets for "true" single sign on*. Newcastle, Storbritannien: Newcastle University.

Gilmore, B., Farvis, K. & Maddock, J. (2004) *Core middleware and shared services studies*. JISC. Tillgänglig på Internet: http://www.jisc.ac.uk/uploaded_documents/CMSS-Gilmore.pdf [Hämtad 2010-02-22].

Gonsalves, A. (2008, 28 Juli) Myspace signs up for single sign-on. *InformationWeek*, 24.

Hodges, J., Howlett, J., Johansson, L. & Morgan, R. L. (2008) *Towards kerberizing web identity and services*. Kerberos consortium. Tillgänglig på Internet: http://www.kerberos.org/software/kerbweb.pdf [Hämtad 2010-03-09].

Jaganathan, K., Zhu, L. & Brezak, J. (2006) *Spnego-based kerberos and ntlm http authentication in microsoft windows*. Internet Engineering Task Force.

Tillgänglig på Internet: http://www.ietf.org/rfc/rfc4559.txt [Hämtad 2010-04-22].

Kouril, D. (2004) *Kerberos module for apache*. Sourceforge. Tillgänglig på Internet: http://modauthkerb.sourceforge.net/ [Hämtad 2010-07-08].

Leng, X. (2009) Smart card applications and security. *Information security technical report, 14*, 36-45.

Linn, J. (2000) *Generic security service application program interface*. Internet Engineering Task Force. Tillgänglig på Internet: http://www.ietf.org/rfc/rfc2743.txt [Hämtad 2010-07-28].

Mather, T. (2002) *Web single sign-on meets business reality*. Sans. Tillgänglig på Internet: http://www.sans.org/reading_room/whitepapers/authentication/web_single_signon_meets_business_reality_130 [Hämtad 2010-02-21].

Moody, G. (2001) *Rebel code: inside linux and the open source revolution*. New York, USA: Basic Books.

Netcraft (2010) *February 2010 web server servey*. Netcraft. Tillgänglig på Internet: http://news.netcraft.com/archives/2010/02/ [Hämtad 2010-02-22].

Netmarketshare (2010) *Operating system market share*. Netmarketshare. Tillgänglig på Internet: http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=10 [Hämtad 2010-02-22].

Neuman, C., Yu, T., Hartman, S. & Raeburn, K. (2005) *The Kerberos network authentication service (V5)*. Internet Engineering Task Force. Tillgänglig på Internet: http://www.ietf.org/rfc/rfc4120.txt [Hämtad 2010-02-20].

Novakov, I. (2006) *Web single sign on systems*. CESNET. Tillgänglig på Internet: http://www.cesnet.asia/doc/techzpravy/2006/web-sso/web-sso.pdf [Hämtad 2010-05-05].

Orton, J. (2008) Kerberos and single sign-on with HTTP. Presenterat vid *ApacheCon Europe '08*, Amsterdam, Nederländerna, 9-11 April, 2008.

Pashalidis, A. & Mitchell, C. J. (2003) A taxonomy of single sign-on systems. I: R. Safavi-Naini & J. Seberry (red:er), *Information security and privacy* (s. 249-264). Heidelberg, Tyskland: Springer-Verlag.

Pashalidis, A. & Mitchell, C. J. (2004) Using emv cards for single sign-on. I: D. Chadwick & G. Zhao (red:er), *Public key infrastructure* (s. 205-217). Heidelberg, Tyskland: Springer-Verlag.

Patnode, M. (2008) Authentication in a heterogeneous environment: integrating linux (and unix and mac) identity management in microsoft active directory. Presenterat vid *Large Installation System Administration Conference*, San Diego, Californien, USA 9-14 november, 2008.

Pfleeger, C. P. & Pfleeger, S. L. (2006) *Security in computing* (4:e upplagan). Upper Saddle River, New Jersey, USA: Prentice Hall.

Scavo, T. & Cantor, S. (2005) *Shibboleth architecture*. Internet2. Tillgänglig på Internet: http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-02.doc [Hämtad 2010-06-28].

Taylor, L. (2002) *Understanding single sign-on*. Intranet Journal. Tillgänglig på Internet: http://www.intranetjournal.com/articles/200205/se_05_28_02a.html [Hämtad 2010-02-21].

Thomas, S. A. (2000) *SSL and TLS essentials: securing the web*. New York, USA: John Wiley & Sons, Inc.

Westman, H. (2010) *Kerberos*. hanswestman.se. Tillgänglig på Internet: http://hanswestman.se/index.php?show=post&id=9 [Hämtad 2010-07-27].

Wilkinson, S. (2006) Kerberizing our network. Presenterat vid *Large Installation Systems Administration*, Durham, North Carolina, USA 21-23 Mars, 2006.

Williams, H. E. & Lane, D. J. (2004) *Web database applications with php and mysql* (2:a upplagan). Sebastopol, Californien, USA: O'Reilly Media, Inc.

Xiong, S. (2005) *Web single sign-on system for WRL company*. Stockholm, Sverige: Royal Institute of Technology (KTH).

Bilaga A – Konfiguration av webbapplikation

När en lyckad autentisering är genomförd med antingen HTTP-autentisering eller Cosign skapas några extra servervariabler för den användarens session. Bland annat skapas *REMOTE_USER eller PHP_AUTH_USER* för PHP och *AUTH_USER* för ASP. De innehåller användarnamnet vilket kan användas för att verifiera vilken användare som loggat in (Williams & Lane, 2004). I tabell 1 och 2 finns exempel på hur användaren kan hanteras i PHP och ASP.

Tabell 2. Exempel på användarhantering i PHP

```
<?php
if( isset( $_SERVER['REMOTE_USER'] ) )
{
    echo("Välkommen ".$_SERVER['PHP_AUTH_USER']."!");
}
else
{
    echo("Du är inte behörig att använda denna sida.");
}
?>
```

Tabell 3. Exempel på användarhantering i ASP

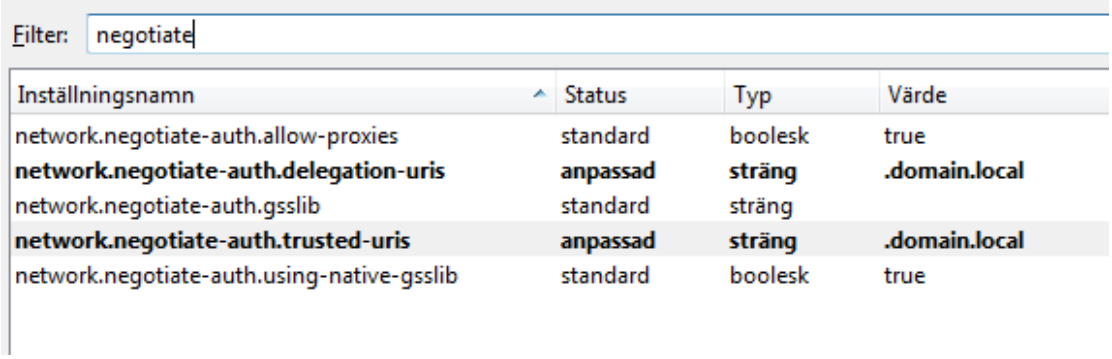
```
<%
If IsNull(Request.ServerVariables("AUTH_USER")) then
    Response.Write("Välkommen " &
Request.ServerVariables("AUTH_USER") & "!")
Else
    Response.Write("Du är inte behörig att använda denna sida.")
End If
%>
```

Bilaga B - Konfiguration av webbläsare

För både Internet Explorer och Mozilla Firefox som är de vanligaste webbläsarna idag krävs det att de konfigureras för att använda HTTP-autentisering. Det behövs för att tillåta dem att använda Kerberos-uppgifter för en viss domän.

Mozilla Firefox

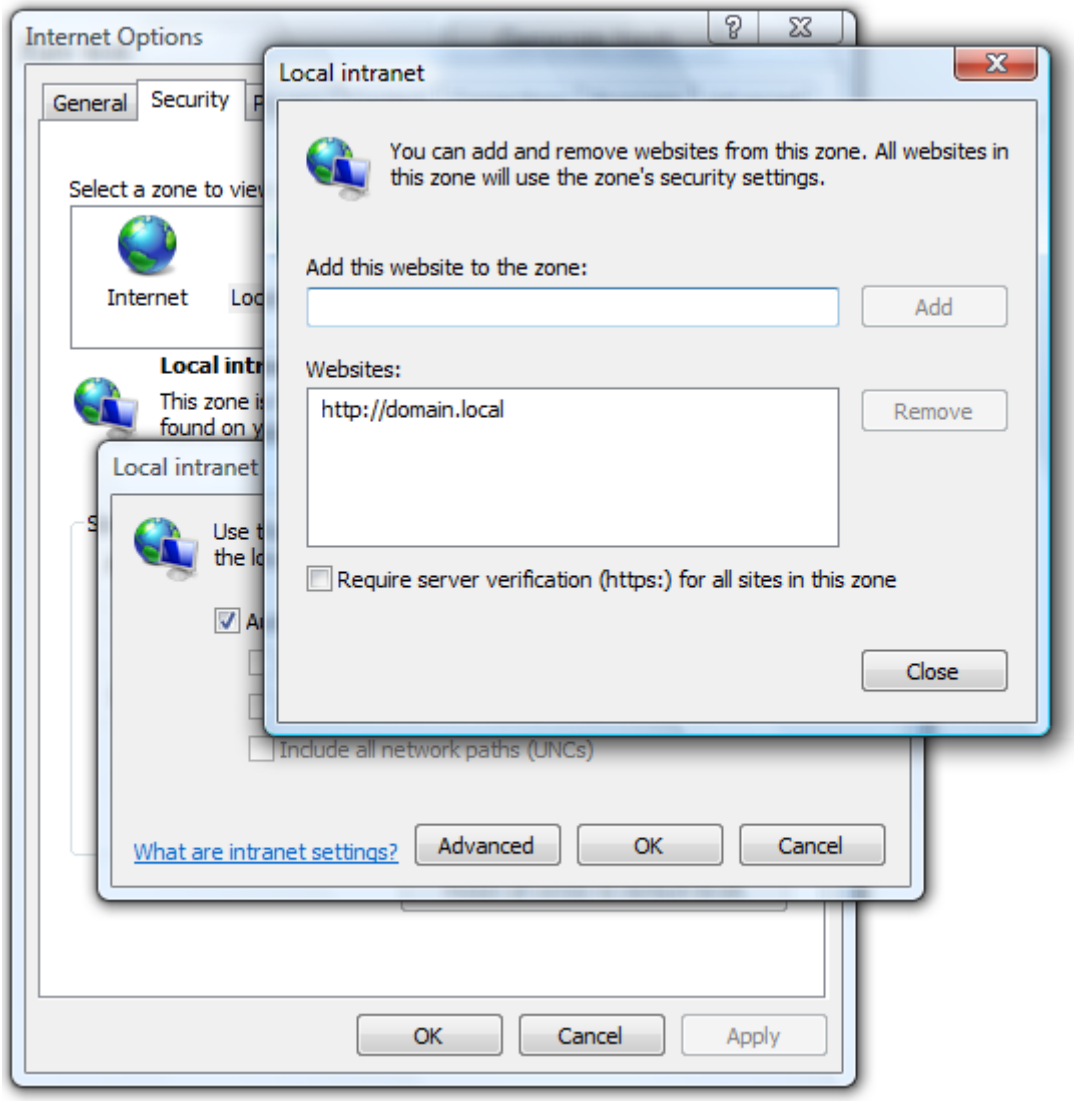
För att konfigurera Mozilla Firefox till att tillåta *NegotiateAuth* måste detta ställas in i webbläsarens konfiguration. Denna nås genom att skriva ”about:config” i adressfältet. För att lättare hitta det relevanta i konfigurationen kan den filtreras på *negotiate*. Detta finns illustrerat i figur 5. Variablerna ” network.negotiate-auth.delegation-uris” och “network.negotiate-auth.trusted-uris” ska ställas in med domännamnet för Kerberos-domänen I formatet “.domain.local”. Då kommer alla webbsidor som tillhör den domänen tillåtas att använda *NegotiateAuth*.



Figur 4. Skärmdump av konfigurationen i Mozilla Firefox

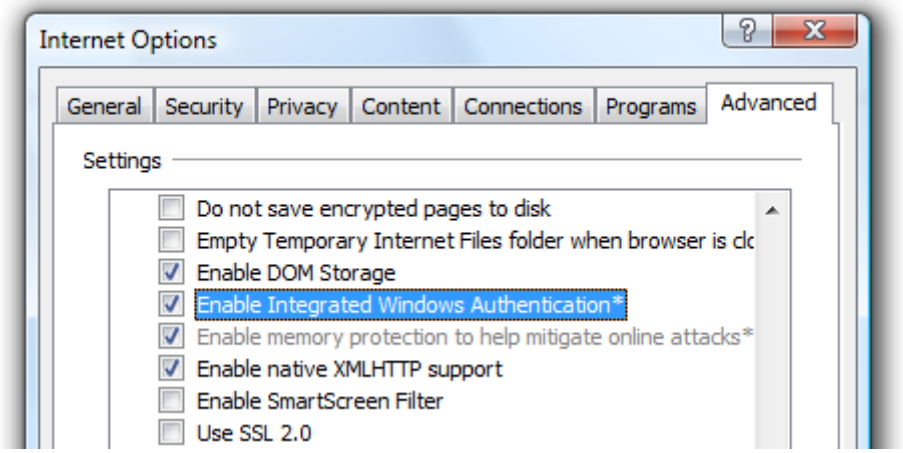
Internet Explorer

För att Internet Explorer ska tillåta en webbplats att använda SSO måste webbplatsen läggas till i en lista med tillåtna adresser. Den listan kan nås genom följande steg i webbläsaren: *Tools* > *Internet Options* > *Security* > *Local Intranet* > *Sites* > *Advanced*. Där kan Kerberos-domänen läggas till. Detta finns illustrerat i figur 6.



Figur 5. Skärmdump från konfigurationen av Internet Explorer

Under: *Tools* > *Internet Options* > *Advanced* kan inställningen ”Enable Integrated Windows Authentication” kontrolleras så den är aktiv för att säkerställa att SSO med *NegotiateAuth* kan användas. Detta finns illustrerat i figur 7.



Figur 6. Andra skärmdumpen av konfigurationen av Internet Explorer