Utvärdering av säkerhet och effektivitet hos CGI och ASP

(HS-IDA-EA-01-110)

Mohamad Mehanna(a98mohme@student.his.se)

Institutionen för datavetenskap
Högskolan i Skövde, Box 408
S-54128 Skövde, SWEDEN

Examensarbete på programmet för systemprogrammering under vårterminen 2001.

Handledare: Nicklas Bergfeldt

Utvärdering av säkerhet och effektivitet hos CGI och ASP

Examensrapport inlämnad av Mohamad Mehanna till Högskolan i Skövde, för Kandidatexamen (B.Sc.) vid Institutionen för Datavetenskap.

01-06-06

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: _____

Utvärdering av säkerhet och effektivitet hos CGI och ASP

Mohamad Mehanna(a98mohme@student.his.se)

Sammanfattning

Internet är det mest populära och kraftfulla nätverksbaserade informationssystem, som erbjuder stora mängder data liggande på servrar runt om på olika platser i världen. Det ständiga uppväxandet av Internet, som framför allt var märkvärdig under de senaste åren, har lett till en informationsrevolution som förmodligen kommer att fortsätta ett antal år framåt i tiden. Databaser som sedan flera år tillbaka varit en grund för lagrings- och sökningsfält är tydligen inte mindre efterfrågat, för dess användning på så väl små som stora företag. Sammanslagning av dessa två världar (Internet och databaser) kommer att ge nya möjligheter för skapandet av avancerade informations-hanteringsapplikationer. Sammankopplingen av databaser på Internet utförs med hjälp av olika tillvägagångssätt som kallas för gateway, och två av dessa är CGI och ASP som studeras i denna rapport med avseende på deras säkerhet och effektivitet. Under genomförandet av arbetet behandlas varje teknologi enskilt. Teknologierna jämförs sedan och detta leder fram till det slutgiltiga resultatet som visar att ASP är bättre lämpat än CGI vad gäller säkerhet och effektivitet.

Nyckelord: Internet, databaser, gateway, CGI, ASP, säkerhet, effektivitet.

Innehållsförteckning

1 Inledning

I företag och organisationer behöver ofta många personer få tillgång till gemensam information, som till exempel dokumentfiler eller schemaprogram (Lindman, 1995). Detta kan till exempel lösas genom att koppla ihop alla berörda datorer i ett nätverk, vilket gör det möjligt för alla att ha tillgång till helt aktuell information. Flera nätverk kan sedan kopplas samman till ett världsomspännande nät som exempelvis Internet (Lindman, 1995).

Internet består av ett antal separata sammankopplade nätverk, vilka är kommersiella, utbildnings- och statsorganisationers nätverk (Connolly & Bygg, 1999). Tjänster som Internet erbjuder är e-post, konferenser, fjärranslutning till andra datorer, filöverföring och chatt-tjänster med mera.

Enligt Ju (1997) är World Wide Web, som består av mjukvara (webbserver och webbläsare) och data (webbsidor), en tjänst eller resurs som lever av Internet. Under de senare åren har databasdesignare börjat märka nyttan av att använda webb-teknologi i kombination med databaser för att öka produktivitet, förenkla informationsspridning och minska papperskostnader i företag, skolor och institutioner. Detta uppnås genom att en webbapplikation sammankopplas med en databas och använder en webbläsare som gränssnitt mot användaren (Ju, 1997).

Ett av de snabbast växande områdena inom Internet är att just koppla databaser till webben (Ek, Arvidsson, & Andersson, 1999). Men vad är det för nytta med att koppla databaser till Internet? Svaret är att det kombinerar det bästa av två världar; Internet och webben är det smidigaste sättet att distribuera information och det fungerar på samma sätt oberoende av dator eller operativsystem som används. Vidare så är databaser bra för att strukturera information och göra den sökbar. Denna kombination är således perfekt för att skapa interaktiva sidor och shoppingsystem, erbjuda säljstöd och mycket annat (Ek, m.fl. 1999).

För att kunna koppla en databas till webben behövs en databas, en webbserver samt mjukvara för att koppla ihop dessa två. För att det ska fungera måste alla ingående komponenter också följa samma standard och idag finns det två vanliga teknologier som används, dessa är CGI och ASP (Ek, m.fl. 1999).

En del viktiga frågor att oroa sig för är förtroendet för dataöverföring och hur effektivt kan data behandlas över webben. När viktiga dokument överförs från webbserver till webbläsare, eller när användaren skickar data tillbaka till servern kan det vara någon som avlyssnar. Avlyssning av information kan ske var som helst på vägen mellan webbläsare och webbserver, även på program som servern använder för att behandla förfrågningar från webbläsaren. Exempel på sådana program är ASP och CGI. Dessutom är det viktigt att den som skickar förfrågan till webbservern får svar snabbt, vilket styrs av effektiviteten hos programvaran som tar hand om informations-förfrågningar på servern (till exempel det CGI- program som körs på servern).

Problemet som undersöks i detta arbete är vilken av teknologierna CGI eller ASP som har bättre förutsättningar vid användning som gateway mellan databaser och Internet med avseende på säkerhet och effektivitet. Arbetet i denna rapport kommer också att visa att CGI och ASP är två viktiga teknologier för webb och databaskommunikation.

Det har visat sig att det inte finns tillgängliga resultat som visar tidigare utvärdering av säkerhet och effektivitet hos CGI och ASP. Det finns däremot flertalet böcker (som till exempel ”Active Server Pages och databaser på Internet”, ”Databases on the

Web”, med mera) som behandlar CGI och ASP var för sig. Dessa tillsammans med praktiskt stöd, i form av kod exempel, har valts som metod för att genomföra detta arbete.

I nästa kapitel diskuteras bakgrunden till Internet, databaser och kopplingen mellan dessa världar, dessutom definieras begreppen som används inom detta sammanhang. I kapitel 3 behandlas problembeskrivning och problempreciseringen mer utförligt. Val av metoder diskuteras i kapitel 4, där presenteras först några allmänna metoder som kan användas, och sedan väljs de metoderna som anses vara lämpliga för utförandet av detta arbete. Kapitel 5 är det viktigaste kapitlet i denna rapport, där gås igenom hur CGI och ASP fungerar och hur dessa behandlar säkerhet och effektivitet var för sig. Det som diskuteras i kapitel 5 analyseras i efterföljande kapitel, och därefter diskuteras slutsatsen i kapitel 7.

2 Bakgrund

Detta kapitel inleds med en kort introduktion om Internet och databaser, sedan kommer begrepp som används vid sammankoppling av dessa två, och som kommer att användas i denna rapport att förklaras. De begrepp som kommer att tas upp är databas, gateway, CGI, ASP, API, skriptspråk, server side includes, http cookies, ODBC, VBScript, JScript och olika tekniker för att förbättra webbservrars funktionalitet (eng. Extending the web server).

2.1 Internet och databaser

Sedan Internet introducerades, har det varit det mest populära och kraftfulla nätverksbaserade informationssystemet, som erbjuder stora mängder data liggande på servrar runt om på olika platser i världen. Det ständiga uppväxandet av Internet, som framför allt var märkvärdig under de senaste åren, har lett till en informations-revolution som förmodligen kommer att fortsätta ett antal år framåt i tiden (Connolly, m.fl. 1999).

Databaser, som sedan 25 år tillbaka varit en grund för lagrings- och sökningsfält (Silberschatz & Korth, 1991), är tydligen inte mindre efterfrågat, inte minst för dess stora användning inom små och stora företag. Sammanslagningen av dessa två världar (Internet och databaser) kommer att ge nya möjligheter för skapandet av avancerade informationshanteringsapplikationer.

Koppling av en databas till webben kräver enligt Ek, m.fl. (1999) en databas, en webbläsare och mjukvara för att koppla ihop dessa två. Mjukvaran är enligt Ju (1997) den gateway som fungerar som en brygga mellan databasen och webbservern.

2.2 Databaser

En databas är en samling av information, som kan innehålla miljontals register (Ju, 1997). Datamängden i en databas är inte relevant, men den måste vara organiserad enligt stränga fördefinierade regler. Databaser kontrolleras via databashanterings-system (DBMSs) som tillhandahåller en metod för manipulering av databasen och data inuti den. Manipulering kan till exempel innebära hämtning av data från databasen, lagring, eller att göra beräkningar på data i databasen. Den populäraste typen av databashanteringssystem som används i den kommersiella världen är relationsdatabashanteringssystem (RDBMS). En annan typ är objektorienterade databashanteringssystem (OODBMS) (Ju, 1997).

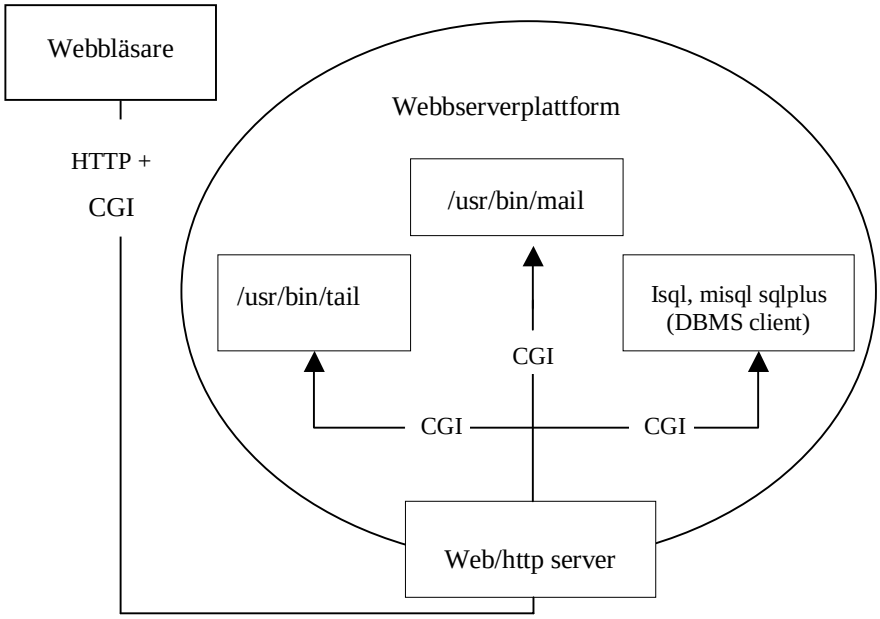
Databaser används för att lagra information på ett strukturerat sätt. Efter lagring av information går det enkelt att söka ut speciell information ur databasen, redigera densamma med mera. Exempel på databaser kan vara ett eget adressregister eller för företag kanske en kunddatabas eller en artikeldatabas över tillhandahållna produkter (Ek, m.fl. 1999).

Databasen kan publiceras statiskt, vilket innebär att när innehållet väl är publicerat så är det inte enkelt att ändra innehållet på webbsidan. Ett annat alternativ är att publicera databasen dynamiskt där informationen lätt kan förändras (Ladd & O'Donnell, 1998). Fördelarna med databaser är flera; dels så går det snabbt och enkelt att hitta den information som behövs genom att söka i databasen, och det är mycket enkelt att ändra på information som redan finns lagrad. Ju mer information som finns lagrad i databasen, desto effektivare används den och desto större blir fördelarna

jämfört med traditionell lagring, som till exempel ren traditionell fillagring. Jämfört med äldre metoder (som till exempel filhanteringsmetoder) finns det större möjligheter med en databas, som att snabbt kunna ändra, söka och sortera efter olika uppgifter (Ek, m.fl. 1999).

2.3 Vad är en Gateway?

En gateway är enligt Ju (1997) en brygga mellan två entiteter. En gateway tillhandahåller en länk mellan världar som är självständiga och inte har något gemensamt så att dessa skall kunna kommunicera med varandra. Avsikten med en databasgateway är att ge en webbaserad applikation förmågan att manipulera data lagrade i databaser. Användandet av gateway och applikationer kallas för att det skapas interaktiva webb-sidor. CGI är ett exempel på ett protokoll som tillhandahåller en gateway mellan webben och program på webbservern (se figur 1). En webbdatabasgateway upprättar således en brygga mellan webben och databaser.



Figur 1: CGI, Common Gateway Interface, fritt från Ju, 1997.

Common Gateway Interface (CGI) är enligt Connolly m.fl. (1999) en av de tidigaste och som fortfarande är den mest använda tekniken för integrering av databaser till webbmiljö. Andra tillvägagångssätt som Connolly, m.fl. (1999) poängterar är följande:

- Server-Side Includes.
- HTTP Cookies.
- Netscape API (NSAPI) och Microsofts Internet Information Server API (ISAPI).
- Java och JDBC (eng. Java Database Connectivity), JSQL och JRB.
- Skriptspråk som JavaScript och VBScript.
- Microsofts Active Platform.

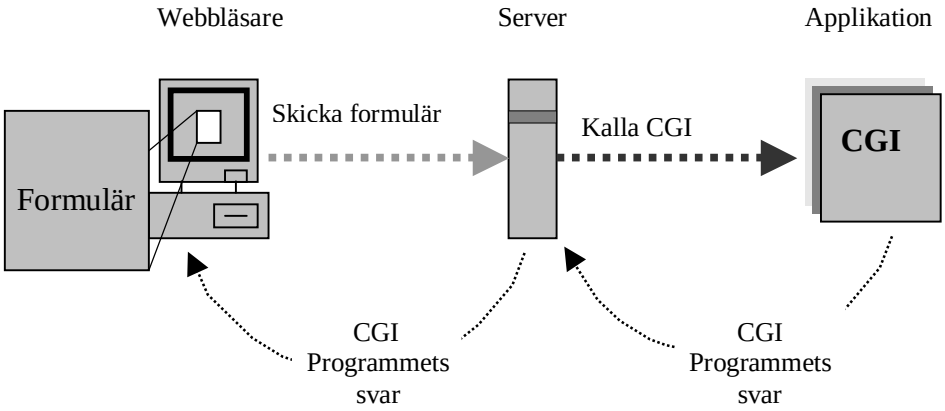
Några av dessa tekniker kommer att förklaras och diskuteras i de kommande avsnitten.

2.3.1 Common Gateway Interface (CGI)

CGI är enligt Ju (1997) ett protokoll som tillåter webbläsarna att skicka förfrågningar till webbservrar och sedan få svar från dessa. Eftersom CGI är ett protokoll och inte ett bibliotek av funktioner specifikt skrivna för något speciellt märke av webbserver, är CGI programspråks-oberoende. Sålänge som programmen rättar sig efter specifikationen för CGI-protokollet, spelar det ingen roll i vilket programmerings-språk programmen är skrivna (det kan till exempel vara C, C++, Perl, eller Java) (Ju, 1999).

När en Internetanvändare arbetar med någon webbsida och fyller i ett formulär på sidan, trycker denne sedan på en knapp för att skicka iväg formulärdata. Data i formuläret packas ihop av webbläsaren och skickas vidare till webbservern via en http-förbindelse. Därefter sker all bearbetning av data i bakgrunden på webbservern.

En webbserver är enligt Ladd, m.fl. (1998) ett program som vet hur att distribuera förfrågningar inkomna från webbläsaren. Webbservrar är inte utvecklade för att bearbeta data som de tar emot från olika formulär, utan det som en webbserver kan göra är att lämna ifrån sig data till ett annat program. Ett sådant program tar hand om data och vet vad det ska göra med informationen. Överlämnandet av informationen från servern till programmet sker med hjälp av Common Gateway Interface (CGI) som motsvarar en uppsättning av standarder med vilka webbservern kan kommunicera med externa program (se figur 2).



Figur 2: Ett enkelt diagram om CGI, fritt från Gundavaram, 1996.

Programmet som bearbetar formulärdata som lämnas från servern kallas för CGI-skript eller CGI-program. Skriptet utför bearbetningar av informationen och kombinerar ett svar som sedan skickas tillbaka till servern via CGI. När servern får svaret skickar servern det vidare till webbläsaren som ställt frågan. Det enda som användaren kan se av hela detta förfarande är formuläret i webbsidan, men både formuläret och CGI behövs för att ställa frågor till och få svar från webbservern.

Sättet hur CGI-program får information från servern beror på servern och dess operativsystem. Alla webbservrar lämnar förfrågningar till CGI tillsammans med så kallade miljövariabler. Sådana variabler innehåller information om format på data i förfrågan, längden på informationen (i byte), användaren som gjorde förfrågan och annan klientinformation. Variablerna innehåller också serverns namn, kommunikationsprotokoll och namnet på mjukvaran som körs på servern. En användarförfrågan kan exempelvis se ut som följer:

```
GET /cgi-bin/welcome.pl http/1.0

Accept: www/source
Accept: text/html
Accept: image/gif
User-Agent: Lynx/2.4 libwww/2.14
From: anvandare@bu.edu
```

Trots många fördelar hos CGI har den sina begränsningar. En av dessa begränsningar är att varje gång ett CGI-program skall köras startar servern en ny process, vilket i sin tur leder till en tung börda på servern vid flera samtidiga förfrågningar (Connolly, m.fl. 1999). För att undvika sådana begränsningar kan andra alternativ till CGI användas, som till exempel Netscapes Server API (NSAPI) och Microsofts Internet Server API (ISAPI) som beskrivs närmare i avsnitt 2.3.2 (Ek, m.fl. 1999).

2.3.2 Extending the web server

För att bli av med de begränsningar som nämndes i föregående avsnitt hos CGI, erbjuder många servrar ett applikationsprogrammeringsgränssnitt (eng. Application Programming Interface, API) som ökar serverns funktionalitet och till och med ändrar serverns beteende. Dessa tillägg kallas för icke- CGI gateway. Två huvud APIer är Netscape server API (NSAPI) och Microsofts Internet Informations Server API (ISAPI). Dessa APIer erbjuder bland annat en metod för att förhindra skapandet av en enskild process för varje CGI skript som skall exekveras (Connolly, m.fl. 1999). APIer kompileras som DLL:er och laddas av webbservern vid uppstarten, och eftersom API-programmen redan finns i servernsminne är de alltid färdiga för att användas, vilket leder till lägre belastning på servern jämfört med användandet av CGI då varje anrop resulterar i nystartande av en process (Ek, Arvidsson & Andersson, 1999).

2.3.3 Server-Side Includes (SSI)

En del webbservrar har möjligheten att analysera dokumenten innan de sänds till webbläsaren, denna process kallas enligt Connolly, m.fl. (1999) för Server- side Include (SSI). Detta till skillnad från att en webbserver vanligtvis inte kontrollerar filer som den sänder till en webbläsare, utan det enda som kontrolleras är om webbläsaren har rätt till att läsa filen.

SSI tillhandahåller kommandon såsom att till exempel tillåta webbdesigners att inkludera aktuell tid och datum i en webbsida, eller att inkludera e-postadress på flertalet hemsidor genom att spara den i en textfil. När e-post adressen sedan behöver ändras räcker det att ändra e-post adressen i textfilen vilket resulterar i att den automatiskt uppdateras på hemsidorna (Ladd, m.fl. 1998).

2.3.4 HTTP Cookies

Cookies är små textfiler som skapas av gateway (CGI, ASP, med mera) på webbservern då klienten besöker en webbsida. Webbservern skickar i sin tur cookien till webbläsaren som sparar cookien på klientens hårddisk. Om klienten (användaren) besöker webbsidan vid senare tillfälle och använder ett exempelvis CGI skript som begär denna cookie, hämtar klienten cookien från hårddisken och skickar informationen i den till webbservern. Användning av cookies gör att CGI-skript blir effektivare då det inte behöver ta emot ny information från samma användare vid flera besökstillfällen av användaren. Informationen hämtas direkt från en cookie som finns på användarens hårddisk (Connolly, m.fl. 1999).

Exempel på hur Cookies fungerar; när någon användare besöker en databas som kräver identifiering av användaren, lagras användarens namn och lösenord i en cookie, som sparas på klientens hårddisk. När klienten besöker databasen igen återkallar CGI skripten cookien från klienten och hämtar tidigare inmatade användarnamn och lösenord så att detta inte behöver skrivas in igen av användaren (Connolly, m.fl. 1999).

2.3.5 Skriptspråk

JavaScript och VBScript är exempel på skriptspråk. Skriptspråk erbjuder inbyggda funktioner och kommandon som tillåter utföring av exempelvis matematiska kalkyler, ändra datasträngar, samt att ta del av och verifiering av data inmatade i ett webbformulär av någon användare (Ladd, m.fl. 1998). Skriptspråk tillåter webbproducenter att skriva små skript som exekveras på användarens webbläsare istället för att exekveras på servern. Till exempel kan en applikation som hämtar data från ett formulär inmatade av någon användare kontrollera om innehållet är komplett och korrekt innan de sänds till webbservern. På så sätt ökas prestanda mellan webbläsaren och webbservern eftersom användare inte behöver sända ofullständiga data till servern i onödan (Ladd, m.fl. 1998).

2.3.6 Active Server Pages (ASP)

ASP står för Active Server Pages och är en teknologi från Microsoft för att göra hemsidor mer levande eller "aktiva" som namnet antyder (Ek, m.fl. 1999). ASP är en lösning som tillåter användandet av något skriptspråk som kan exekveras på servern. Eftersom det tillåter skripten att exekveras på servern får det tillgång till serverns resurser, som till exempel databaser, som annars är helt osynliga för klienten. ASP tillhör Microsoft Active Platform som beskrivs nedan.

Microsoft Active Platform är en ”öppen standardbaserad mjukvaruarkitektur för utdelning av applikationer över Internet och intranet.” (Connolly, m.fl. 1999, s. 891). Microsoft Active Platform innehåller olika verktyg, tjänster och teknologier såsom HTML, skript (som till exempel JavaScript, VBScript och andra skriptspråk) och komponenter (Java eller ActivX[1]) (Connolly, m.fl. 1999).

Alla ASP filer slutar med .asp, och sådana filer kan innehålla en kombination av text, HTML-taggar som avgränsas med speciella tecknen (< och >), samt skript-kommandon och output uttryck som avgränsas med tecknen (% och %) (Connolly, m.fl. 1999). Rent praktiskt fungerar ASP som en sorts HTML-generator där allt innanför avgränsarna processas och eventuellt omvandlas till HTML-kod, och allt utanför skickas till webbläsaren som det är (Ek, m.fl. 1999).

En ASP-sida på Internet kan innehålla olika skriptspråk, och det vanligaste språket är VBScript som är baserat på Visual Basic. Visual Basic-programmerare (samt gamla BASIC-programmerare) har alltså mycket gratis här, vilket är en bidragande orsak till att ASP har blivit populär på kort tid (Ek, m.fl. 1999).

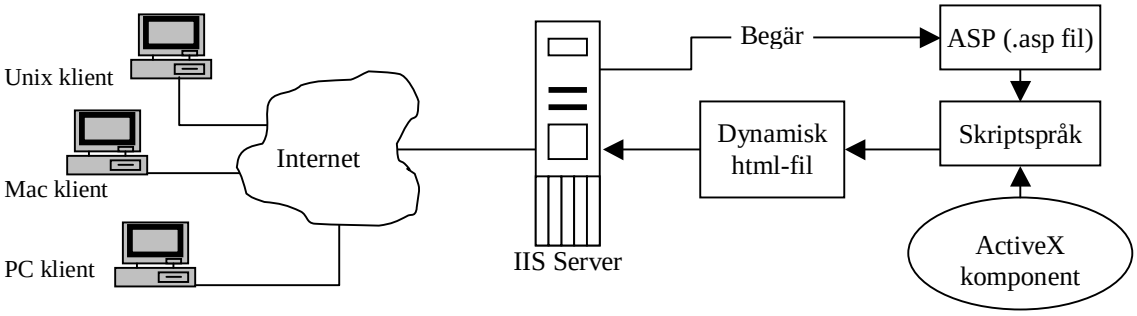
VBScript har också inbyggt stöd för databashantering, vilket gör att det tillsammans med SQL-anrop dessutom lämpar sig bra att användas tillsammans med databaser.

[1] ActiveX är Microsofts standard för interaktivitet på WWW. En teknik för att inkludera olika saker på en WWW-sida såsom till exempel ljud, bilder, knappar, med mera.

Den stora fördelen med ASP är att det går att använda andra skriptspråk, som till exempel JavaScript (Ek, m.fl. 1999).

Skripten kan antigen köras på klienten (webbläsaren) eller på servern. Genom att köra ASP på klienten avlastas servern något vilket är värdefullt vid stora webbplatser där flera användare besöker platsen samtidigt. Men alla klienter har inte stöd för skript, och framför allt inte för olika typer av skript, och detta leder till att informationen inte blir tillgänglig för alla användare. Genom att köra skript på servern fungerar det på vilken klient som helst. En process som körs på servern kan komma åt serverns resurser, men nackdelen med att köra skript på servern är att den kan bli tungt belastad vid hantering av stora mängder av samtidiga besök (Ek, m.fl. 1999).

ASP skriptet startar körningen när webbläsaren begär en .asp fil från webbservern. Webbservern anropar sedan ASP som läser igenom filen från början till slut med exekvering för alla kommandon som hittas. Sedan skickas den genererade HTML-filen (sidan) till webbläsaren (se figur 3) (Connolly, m.fl. 1999).



Figur 3: Active Server Pages. Fritt från Connolly m.fl. 1999

2.3.7 ODBC

ODBC står för ”Open Database Connection” och är en standard framtagen av Microsoft för att arbeta mot databaser. ODBC- gränssnittet hanterar data lagrad i en databashanterare (DBMS) med språket SQL. Genom att använda standardmetoder som ODBC, behöver de som gör sina applikationer tillgängliga på Internet inte tänka på vilken databashanterare eller vilket nätverk som deras applikationer kommer att köras på. Detta gör naturligtvis dessa applikationer tillgängligt för så många fler användare och miljöer. Det är således fullt möjligt att utveckla och kompilera ett program utan att i förväg veta vilken databashanterare som kommer att användas (Ek, m.fl. 1999).

ODBC kan enligt Ek, m.fl. (1999) betraktas som ”limmet” mellan webbservern och databasen som stödjer både webbserverns gränssnitt (CGI, NSAPI eller ISAPI) och databasens gränssnitt. ODBC kan alltså användas som ett lager på toppen av databasen så att den tar emot förfrågningar från gateway och skickar dessa vidare till databasen oavsett vilken typ databasen är av.

2.4 VBScript

VBScript är ett av skriptspråken som används i ASP. Fördelen med att använda ett sådant skript kan förklaras med följande exempel; en användare surfar på Internet och hittar en webbsida som innehåller vissa erbjudanden. Användaren väljer att fylla i

formuläret i sidan och klickar sedan på sänd knappen för att skicka formuläret över Internet till servern. Servern upptäcker att surfaren glömt fylla i någon av uppgifterna i formuläret och skickar ett meddelande tillbaka så att surfaren får backa ett steg och fylla i rätt värden, för att sedan skicka formuläret igen. Det är mycket möjligt att denna procedur upprepas ett flertal gånger och det är sånt som tar tid och förmodligen bara de mest envisa står ut med detta och väntar.

En möjlig lösning är att använda ett skriptspråk som till exempel VBScript, som gör att formuläret rättas på surfarens dator med dess egen kraft, istället för att belasta nätet och servern. Denna process leder framför allt till tidsvinst och prestanda i form av minskad nätverksbelastning och minskad belastning på servern.

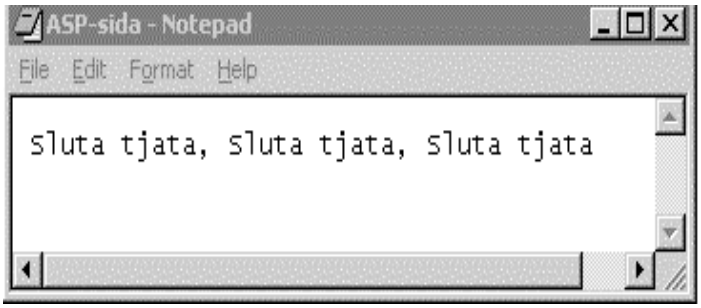
2.5 JScript

JScript är ett kraftfullt objektorienterat språk och har ett antal objekt vars metoder kan underlätta arbetet med en webbsida. Genom att använda JScript i webbsidan blir den tillgänglig till flera surfare. (till skillnad från VBScript, där sidan bara blir tillgänglig för användare som använder Internet Explorer). JScript är ett av de officiella ASP-språken som kan användas utan att behöva extra skript-tolk. För att ställa in ett JScript för en enskild sida skrivs exempelvis:

```
<%@LANGUAGE = JScript%>

<HTML>
<HEAD> <TITLE> ASP-sida </TITLE> </HEAD>
<BODY>
<var I
while (i<3) {%>
    sluta tjata.
<% i++ }%>
</BODY>
</HTML>
```

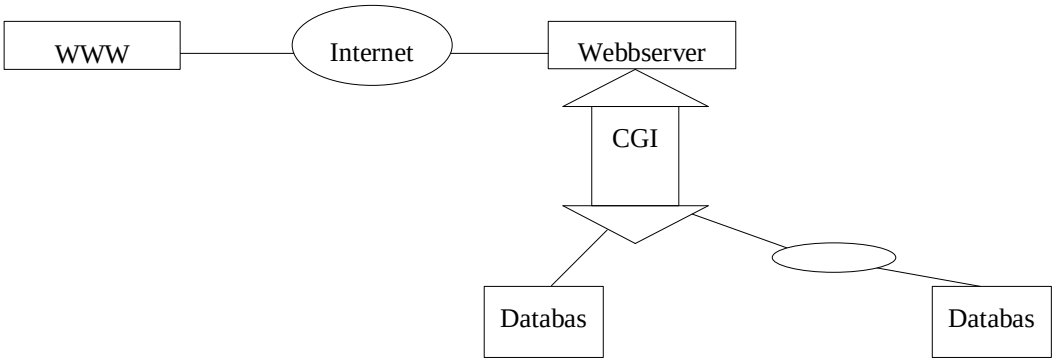
Som efter exekvering ser ut som i figur 4.



Figur 4: Resultat av exekvering av JScript kod. Fritt från Ek, m.fl. 1999.

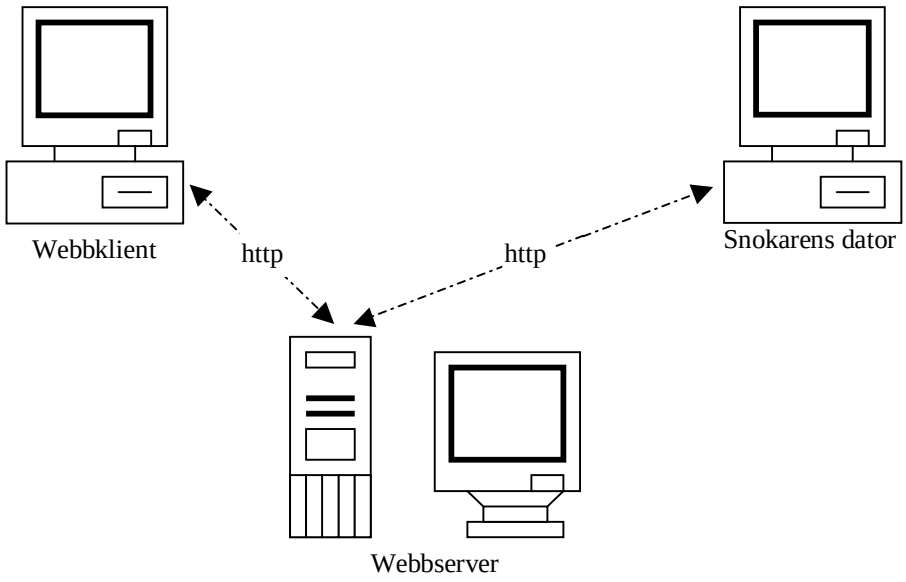
3 Problembeskrivning

När en webbanvändare begär en sida som behöver information från en databas krävs det flera steg för att fullfölja denna begäran (Ladd, m.fl. 1998). Webbservern tar emot förfrågan från webbplatsen och skickar informationen vidare till ett skript (till exempel ett CGI-skript). Detta skript fungerar som en bro som förbinder två olika system. Skriptet utför själva frågeställningen, tar emot resultatet från databasen, formulerar ett lämpligt svar och sänder det till webbservern som i sin tur vidarebefordrar det till användarens webbläsare (se figur 5).



Figur 5: Informationsflödet mellan webben och databaser, fritt från Ladd, m.fl. 1998.

Sådana förfrågningar och webbdatabasapplikationer som överförs mellan webbläsare och webbserver över Internet kräver högre säkerhetsnivå jämfört med applikationer transporterade över exempelvis intranet (Ju, 1997). Information som sänds över Internet kan därför vara osäker, vilket innebär att förfrågningar som sänds mellan webbläsarna och webbservrar kan avlyssnas av en tredje part (se figur 6).



Figur 6: En osäker webblänk kan avlyssnas, fritt från Ju 1997.

Effektiviteten är ju givetvis inte mindre viktig, för när en användare begär något från

servern ska han/hon inte behöva vänta länge innan servern skickar svaret till webbläsaren som används av användaren.

Enligt Ju (1997) består webbdatabasapplikationer av fyra komponenter eller lager (skikt), dessa är webbläsaren, applikationslogiklager, databaskopplingslager och ett databaslager. Transmitionssäkerhet och frågehanteringseffektivitet hanteras av applikationslogiken som kan existera som en gateway (CGI, server API program, med mera) på webbservern. Applikationslogiken tar hand om dataanskaffning för en förfrågan (exempelvis, en SQL sats). Applikationslogiken kan också, enligt beskrivningen ovan, förbereda förfrågan och skicka den till databasen, för att sedan hämta resultaten och göra de tillgängliga för webbläsaren. Hur effektivt och säkert detta utförs varierar beroende på vilken gateway som används.

Problemet med att ge användare på webben tillgång till en databas är att databasprogrammet litar blint på gateway-skriptet som används i webbservern (till exempel ett CGI-skript). Detta kan leda till säkerhetsproblem om någon med onda avsikter får tillgång till ett skript som har möjlighet att ändra i databasen.

3.1 Problemprecisering

Som nämnts i föregående avsnitt, är säkerheten och frågehanteringseffektiviteten vid webbdatabaskommunikation en viktig uppgift som en gateway måste hantera tillfredsställande. Med säkerhet menas att information som behandlas av teknologin skall kunna skyddas så att främmande inte får komma åt denna information, och även hur säkert denna information behandlas av teknologin, med tanke på om det finns risk för att information förstörs. Med effektivitet menas hur snabbt behandlas förfrågan av teknologin från det att förfrågan tagits emot tills det att svaret lämnats tillbaka till webbservern. Från denna utgångspunkt, och för säkerhetens och effektivitetens viktighet vid överföring av information mellan webbläsare och webbserver över Internet, kommer arbetet i denna rapport att utvärdera skillnaderna mellan CGI och ASP med avseende på säkerhet och effektivitet. Resultatet av arbetet kommer att underlätta för personer eller företag som vill koppla sina databaser till Internet att enkelt välja typen av gateway som skall användas för att göra sina databaser tillgängliga till Internetanvändare på ett säkert och effektivt sätt.

3.2 Motivering

Fokus på detta arbete kommer att ligga på utvärdering av säkerhet och effektivitet hos CGI och ASP. CGI är en standard (teknologi) som har funnits sedan början av samman-slagningen av databaser och Internet och betraktas av flera författare som en viktig standard vid koppling av databaser till Internet. Dessutom är CGI programmeringsspråksoberoende och kan programmeras i vilket programmerings-språk som helst. ASP är en standard som tillåter användning av olika skriptspråk med inbyggt stöd för databashantering som sedan exekveras på servern och får tillgång till serverns resurser (som till exempel databaser). Sådana egenskaper finns inte hos många andra gateways, vilket gör att ASP föredras av såväl Visual Basic- som Java-programmerare.

3.3 Förväntat resultat

Enligt Ju (1997) är CGI standardsättet för att komma åt program på en webbserver från en webbläsare, och att CGI kommer att fortsätta vara den främst använda gateway för att koppla databaser till webben.

Efter utförandet av denna rapport förväntas resultat som visar att Active Server Pages (ASP) är en annan gateway, eller teknologi, med förbättrad effektivitet och säkerhet gentemot CGI vid informationstransmission mellan databaser och Internet.

4 Metoder

I detta kapitel presenteras olika metoder som kan användas för att utvärdera och lösa problem i form av det som behandlas i denna rapport. Sedan görs ett metodval för den metoden som anses lösa det beskrivna problemet i föregående kapitel (kapitel 3.1) och som kan arbeta fram det slutgiltiga resultatet som kommer att visa om det förväntade resultatet uppnås. Vid valet görs motivering om varför den valda metoden (eller metoderna) har valts, och eventuella för- och nackdelar vid användandet av dessa metoder.

4.1 Litteraturstudie

Litteraturstudie innebär bland annat att undersökaren använder sig av befintliga dokument. Med befintliga dokument avses vanligen det som trycks eller nedtecknas såsom böcker och artiklar. Begreppet dokument kan också användas för filmer, fotografier och bandspelningar (Patel & Davidsson, 1994). Metoden litteraturstudie används oftast för att besvara eller belysa läsarens frågeställningar kring fakta och exempelvis undersökning av ett problemområde.

Litteraturstudie är en bra metod för att exempelvis undersöka egenskaper hos olika teknologier, för att sedan jämföra dessa och komma fram till den slutsatsen som visar vilken av teknologierna har bättre egenskaper, särskilt om litteraturen (böcker, artiklar, med mera) är skriven av olika författare. Exempel på arbete som kan få nytta av denna metod är arbetet som behandlas i denna rapport. Detta för att det finns en mängd litteratur i form av böcker och artiklar som behandlar CGI och ASP i separata sammanhang, vilket ger möjligheten att jämföra det som skrivs på böcker med det som kontinuerligt publiceras på artiklar. Jämförelsen kan göras för hur dessa teknologier fungerar vid behandling av förfrågningar med tanke på säkerhet och effektivitet.

Fördelen med att använda litteraturstudie är tillgängligheten av information inom båda teknologierna (CGI och ASP), och särskilt när det gäller artiklar som ständigt publiceras med färsk information om erfarenheter kring dessa teknologier. En nackdel med böcker är att böcker brukar vara dyra att köpa, vilket gör att läsaren antingen blir tvungen att bekosta sig och köpa det som behövs, eller att vända sig till biblioteken och låna. Ibland är böckerna redan utlånade och blir inte tillgängliga igen på upp till en månad vilket ytterligare kan försvåra processen.

4.2 Intervjuer

Intervjuer är en annan metod som kan användas för att samla information. Denna metod används för att öka förståelsen för det som läses i litteraturen så att undersökaren kan jämföra det som läses med hur det fungerar i verkligheten, eller att söka fakta som inte kunnat hittas med litteraturläsning. Intervju kan ske på två sätt; antingen via telefon eller via besök. Besöksintervju tar längre tid och kräver att frågeställaren måste resa till den som skall intervjuas, vilket gör att intervjun blir dyr att genomföra. Telefonintervju är snabbare och brukar inte vara lika kostsam jämfört med intervjuer utförda via besök.

För att kunna utföra arbetet i denna rapport med hjälp av denna metod kan ett antal frågor om säkerhet och effektivitet hos CGI och ASP formuleras. Genom att sedan besöka, eller ringa folk med kunskaper och erfarenheter kring dessa teknologier, ställs

frågorna, och de erhållna svaren kombineras sedan med information från litteratur, för att bearbeta slutsatsen som visar vilka av dessa teknologier som är säkrare eller effektivare.

Fördelen med denna metod är att svaren kan fås från personer med kunskaper och erfarenhet inom CGI och ASP. Men denna typ av folk anses inte vara lätt att få tag i inom den tiden som är avsedd för utförandet av arbetet i denna rapport, vilket gör att denna metod inte är lämplig.

4.3 Enkäter

Enkäter är också ett vanligt alternativ som väljs för att göra någon typ av undersökning. Med enkätmetoden konstrueras ett antal frågor med olika svarsalternativ som sedan skickas till personer och företag som av undersökaren bedöms vara insatta i det område som undersöks, för att fylla i och sedan skickas tillbaka. När undersökaren får tillbaka svaren gör denne jämförelser mellan de olika erhållna svaren och drar slutsatser enligt dessa.

Enligt Patel & Davidsson, (1994) kan enkäter antingen genomföras genom att frågeställaren träffar respondenten, eller genom att skicka enkäten via vanligt post eller e-post. Enkätmetoden kan användas för arbetet i denna rapport genom att formulera frågor enligt föregående avsnitt, och för varje fråga ge ett antal alternativa svar som respondenten kan använda sig av för att kunna svara på respektive fråga. När enkäterna fås tillbaks, kombineras svaren och utifrån dessa dras slutsatsen om vilken teknologi som är säkrare och effektivare. Nackdelen med denna metod är att den kan ta lång tid innan enkäterna skickas tillbaks, eller att mottagaren helt struntar i enkäten, vilket kan försena utförandet av arbetet. Därför bedöms denna metod som tidskrävande och ej lämplig för utförandet av arbetet i denna rapport.

4.4 Empiriska studier

Implementation är ett alternativ på metod som kan användas för att lösa vissa problem. Metoden går ut på att undersökaren implementerar en applikation som kan sedan köras och testas på de olika undersökta teknologierna, och utifrån detta drar undersökaren den slutgiltiga slutsatsen om vilka av dessa teknologier som bäst hanterar applikationen.

Det bästa resultatet som förmodligen kan uppnås är genom att implementera en applikation som körs med CGI och ASP var för sig, och utifrån denna mäta säkerheten och effektiviteten hos de båda. Ett problem med ett sådant arbete är att det kan ta lång tid att genomföra eftersom det handlar om dubbla utvecklingsprocesser. Ett annat problem är att testaren måste ha systemadministratörsrättigheter för att köra ett CGI-program, eftersom dessa program måste lagras i ett bibliotek på servern som bara kan hanteras av systemadministratören.

4.5 Val av metod

Vid val av metod skall den metoden som anses passa bäst för utförandet av arbetet väljas. Bland de ovan nämnda metoderna anses litteraturstudie den mest passande metoden som kan användas för att komma fram till det slutgiltiga resultatet för det beskrivna problemet i kapitel 3.1.

Anledningen till att litteraturstudie har valts är att det finns ett antal olika författare som har skrivit böcker om CGI och ASP och om hur dessa teknologier fungerar.

Litteratur som finns idag tar upp dessa två var för sig, och genom att läsa böcker, eller artiklar, skrivna av olika författare inom varje område kan sedan den erhållna informationen jämföras och kombineras för att bilda en god uppfattning om respektive område. Några av de valda böckerna som kommer att användas vid samling av information om CGI och ASP är:

- "Databases on the web" (Ju, 1997). Boken beskriver kopplingen mellen databaser och Internet med fyra steg och tar även upp säkerhetsaspekter som används på nätet. Författaren medverkar även i ett antal olika böcker inom området.
- "Database Systems. A practical approach to design, implementation, and management" (Connolly m.fl. 1999). En bok som behandlar olika tillvägagångssätt mellan databaser och Internet, och beskriver dessutom olika typer av databaser.
- "Active Server Pages och databaser på Internet" (Ek, m.fl. 1999). Författarna har skrivit ett antal böcker inom ASP, och översätter det nyaste inom ASP från engelska till svenska.
- "HTML, JAVA och CGI" (Ladd, m.fl. 1998). Boken innehåller beskrivningar om CGI, och några andra tillvägagångssätt som SSI, VBScript, med mera. Författarna har skrivit ett antal olika böcker inom området.
- "CGI Programming on the world wide webb" (Gundavaram, 1996). Författaren har även skrivit andra böcker om CGI och medverkar dessutom med andra författare.
- "ASP 3.0, Programmer's Reference" (Anderson m.fl. 2000). Boken behandlar endast ASP 3.0 och är skriven av tio författare. Författarna, tillsammans med några andra författare, har även skrivit en upplaga av boken på svenska.

Exempel på artiklar som kan användas är den veckopublicerade nyhetsbrevet "Active news" som innehåller diskussioner och erfarenheter om bland annat CGI och ASP.

För författare, med mera hänvisas till referenslistan i slutet på rapporten.

5 Genomförande

I detta kapitel beskrivs CGI- och ASP- teknologierna utförligt, och beskrivningen utförs för varje teknologi i ett separat avsnitt. Avsnitten börjar med en generell bild av hur dessa teknologier fungerar, och sedan poängteras effektiviteten och säkerheten hos de båda. Det första avsnittet i kapitlet behandlar CGI, och det efterföljande avsnittet behandlar ASP.

5.1 Common Gateway Interface (CGI)

CGI är det vanligaste programmeringsgränssnittet för Internet-servrar, som kan ta emot förfrågningar från webbservern och formulera svar som sedan skickas till webbläsaren (Ek, m.fl. 1999). Common Gateway Interface är en del av webbservern som kan kommunicera med andra program liggande på servern. CGI fungerar som en bro som förbinder två olika system. (Gundavaram, 1996).

5.1.1 CGI och Webbservern

Kommunikationen mellan klienten och exempelvis en databasserver måste alltid ske via webbservern. För varje förfrågan från klienten, och för varje svar från databasservern, måste webbservern konvertera data från och till html format i form av html-dokument. Vid flera förfrågningar till webbservern från flera användare samtidigt, blir det trängsel i kommunikationen, vilket belastar servern och försämrar dess prestanda och möjligheten att behandla ytterligare inkommande förfrågningar (Connolly, m.fl. 1999).

Innan servern startar något CGI-program skapar den ett antal systemvariabler som indikerar serverns status och dessa lämnas sedan till CGI-skripet. Varje skript får en unik uppsättning variabler som sedan försvinner när skriptet blir klart. En aktiv server kan ha flera skript som körs parallellt, och var och ett körs med sin egen miljö (Ladd, m.fl. 1998).

När CGI-programmet startar skapar det antingen en ny fil och skickar dess output till servern i form av en dataström, eller lämnar URL:en till en fil som redan finns (Gundavaram, 1996). Utöver att sända egen utdata är skriptet ansvarig för att skapa information i form av MIME[2]-kod, som skickas sedan vidare från servern till webbläsaren. Denna kod talar om för webbläsaren vilken typ av fil som är på väg över nätet (filen kan till exempel innehålla text, bilder, ljud, med mera). Eftersom MIME-koden skickas före filen kallas den för header-information.

Huvudet kan antingen skickas som en del-http huvud (eng. partial header) eller full-http huvud. Är det ett del http-huvud skickas informationen till servern som i sin tur skickar den vidare till webbläsaren som ställde förfrågan. Annars skickas informationen direkt till klienten utan serverns inblandning (Gundavaram, 1996).

Webbservern måste generera en ny process, eller tråd för varje CGI-skript, för att skripten körs som separata program på servern. Vid exekvering av ett antal skript samtidigt börjar processerna tävla om att få tillgång till serverns resurser (som till exempel minne, disk och processortid) vilket lägger extra arbete på webbservern och

[2] MIME (Multipurpose Internet Mail Extensions) är en specification som utvecklades från början för att kunna skicka olika typer av data genom e-post. MIME typer används för att identifiera innehållet av ett dokument som skickas över webben.

tar upp stor del av dess processorkraft (Connolly, m.fl. 1999). Troligen är det inga problem för servern att hantera några fåtal uppgifter samtidigt, men ibland kan det handla om hundratals förfrågningar (URL-förfrågningar). Om många processer (en process för varje förfrågan) körs samtidigt kan detta överbelasta servern och leda till långa svarstider, medan användaren sitter på andra sidan och stirrar på bildskärmen i väntan på att något skall dyka upp (Ladd, m.fl. 1998).

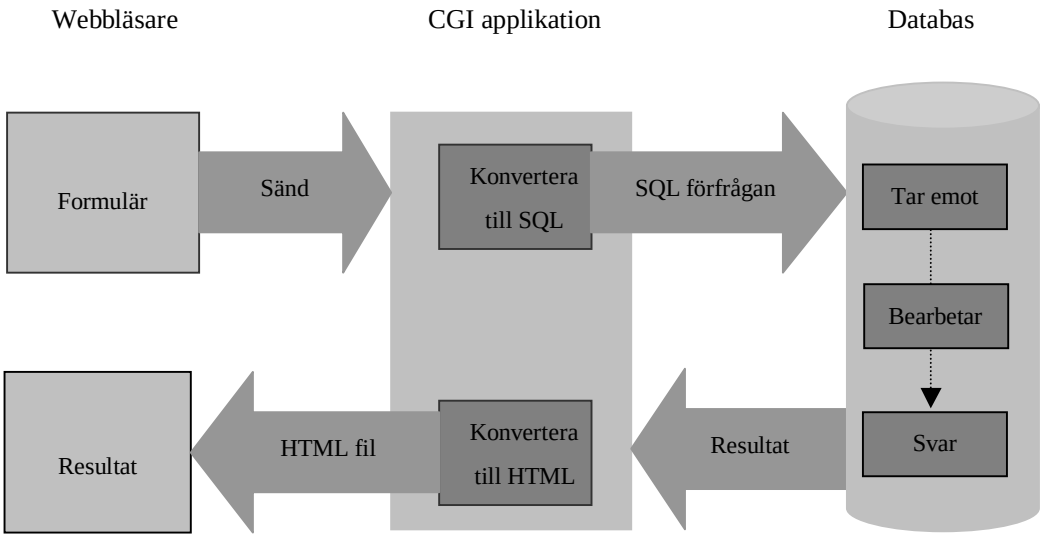
Ibland kan flera besökare begära en och samma process samtidigt (ett gästbok skript kan till exempel visa innehållet för tre besökare, samtidigt som den tar emot indata från en fjärde besökare). Det tar tid för servern att låsa en fil när den skrivs och öppna den när den har skrivits färdigt, och att skapa ett skript som kan klara det gör själva skriptet komplicerat, större och mer invecklat (Ladd, m.fl. 1998).

5.1.2 CGI och Databaser

Det finns inga begränsningar för vad som kan kopplas till webben, men som ett enkelt exempel; ett företag vill koppla upp dess stora databas till Internet och göra den tillgänglig till alla Internet användare världen över. Företagets databas innehåller bland annat tredimensions data (grafik) och multimedia. Allt som behövs för att utföra en sådan uppgift är att skriva ett CGI-program som utför följande tjänster (Gundavaram, 1996):

1) Omvandlar inputdata,
2) Skapar förfrågningar av någon sort,
3) Skickar förfrågningarna till databasen,
4) Bearbetar det returnerade resultatet från databasen,
5) Skickar ett html-dokument till användaren.

Detta genomförande förklaras ytterligare med hjälp av figur 7. För varje förfrågan



Figur 7: CGI till databaser, fritt från Gundavaram, 1996.

som CGI lämnar till databasservern utför denna server samma input- och output-procedur, som om det hade varit separata förfrågningar från olika källor. Anledningen är att servern måste veta klientens identitet varje gång den tar emot en förfrågan, även om förfrågningarna hör till en och samma användare. Orsaken är tillståndslösheten

hos http-protokollet som avbryter kommunikationen efter skickande och emottagning av varje förfrågan (Ladd, m.fl. 1998).

Eftersom CGI-program är exekverbara, kan det innebära en risk att tillåta alla Internetanvändare världen över att komma in i företagets system, eftersom det kan leda till säkerhetsproblem. Därför måste några försiktighetsaspekter tas i beräkning vid koppling av databaser till Internet och särskilt när andra får tillgång till informationskällan (databasen) på företaget.

5.1.3 Försiktighetsaspekter

Det vanligaste sättet att identifiera databasanvändare och ge tillgång till informationen i databasen är att kunna identifiera användare genom att de får knappa in dess användarnamn och lösenord (Ju, 1997). Ett annat sätt att kontrollera användarens tillträde till informationskällan är genom att kontrollera användarens IP-adress som finns lagrad i CGI:s REMOTE_HOST[3] miljövariabel. Men att identifiera användare genom detta faller snart på olika sätt. Till exempel kommer REMOTE_HOST variabeln inte att kunna skilja mellan miljontals IP-adresser för användare som samtidigt kopplar upp sig på en webbplats. Eller, genom att en kunnig användare som kan inbilla servern att betrakta dennes dator som någon annans IP-adress (Ju, 1997).

CGI-program måste placeras i en speciell katalog på webbservern så att servern exekverar programmen istället för att skicka och visa de på webbläsaren. Det främsta orsaken för att lägga alla CGI-program i en katalog på servern är att systemadministratören får direkt kontroll över hela systemet, och över alla program som körs på servern. Vilket ökar säkerheten i systemet genom att bland annat förbjuda användare från att köra egna CGI- program (Gundavaram, 1996).

Däremot finns det direktiv som tillåter sådana program att exekvera utanför en speciell katalog, baserad på filändelser (till exempel filer med extensionen .pl eller .cgi) vilket är farligt för säkerheten i systemet. Farligheten ligger i att användare kan skriva egna skript och köra dessa på servern, vilket kan påverka säkerheten i systemet. Ett exempel är användare som skriver ett eget skript som har till uppgift att skriva ut innehållet av en viktig fil/filer till variabeln som lämnar information från CGI till servern (standard OUTPUT) så att de sedan för vidare innehållet från servern (Gundavaram, 1996). Inmatning av data från användare i formulär måste därför kontrolleras och undersökas när den tas emot till CGI-programmet. Kapitlet 5.1.4 beaktar ett antal exempel på hur säkerheten kan skyddas i CGI (Gundavaram, 1996).

5.1.4 Säkerhetsåtgärder

Vid inmatning av data i ett formulär finns det risk för att någon användare använder sig av sådana tecken som kallas för ”shell metatecken” i form av data. Sådana tecken har speciell betydelse för skalet (eng. shell) och kan påverka systemets beteende. Processen förklaras nedan och börjar när användaren blir tillfrågad att mata in sitt namn i ett formulär:

```
<FORM ACTION=”/cgi-bin/finger.pl” METHOD=”POST”>
<INPUT TYPE=”text” NAME=”user” SIZE=40>
```

[3] REMOTE_HOST: är en av CGIs miljövariabler som hanterar fjärrdatornamn för användare som gör förfrågningar. Information om denna variabel innehåller adressen till en enda användare, eller en lista med adresser till flera användare, saknas i böckerna som behandlar denna.

```
<INPUT TYPE=”submit” VALUE=”Get Information” >
</FORM>
```

Programmet som behandlar inmatningen kan se ut som följer

```
#!/usr/local/bin/perl
&parse_form_data(*simple);

$user = $simple(‘user’);
print “Content-type: text/plain”, “\n\n;
print “Här är resultatet på din förfrågan: ”, ”\n”
print ’/usr/local/bin/finger $user’;

print “\n”
exit (0);
```

Detta är ett exempel på ett mycket farligt program vilket helt bör undvikas. Programmet kan ta emot och exekvera vilken data som helst från användaren. Tänk om användaren matar in följande tecken som användarnamn (Gundavaram, 1996):

; rm * ; mail –s ”Då så” forstora@crack.net < /etc/passwd

Denna sekvens kommer att ta bort alla filer i katalogen och även maila, /etc/passwd, filen som finns på systemet till den berörda användaren. För att undvika sådana problem måste data kontrolleras innan den tas emot och exekveras. Programmet ovan kan till exempel modifieras till:

```
#!/usr/local/bin/perl

&parse_form_data(*simple);
$user = $simple(‘user’);

if ($user =~ / [ ; < > & \ * ’ \ |] / ) {
  &return_error (500, “CGI Program Alert”, “ Vad försöker du göra?”);
  }
else {
  print “Content-type: text/plain”, “\n\n;
  print “Här är resultatet på din förfrågan: ”, ”\n”
  print ’/usr/local/bin/finger $user’;

  print “\n”
  } exit (0);
```

Genom att modifiera programmet så att det ser ut som ovan blir det säkrare eftersom all data som matas in kontrolleras efter specifika tecken, som till exempel; ” , ” , ” > ” , ” < ” , ” & ” , ” * ” , ” ’ ” och ” | ”. Om informationen innehåller något av dessa tecknen returneras ett felmeddelande till användaren, annars returneras resultatet på den inmatade informationen (Gundavaram, 1996).

5.1.5 Sammanfattning

CGI är en teknologi som gör att två olika system kan kommunicera med varandra. Det är en del av webbservern som servern använder för att ställa frågor och få svar till och från program som har koppling till denna. Servern tar emot förfrågningar från webbläsaren (klienten) och lämnar de vidare till CGI-programmet som tar hand om dessa. Ett CGI-skript (eller CGI-program) startar och börjar behandla förfrågan genom att formulera någon typ av förfrågan som det i sin tur ställer till destinationen (till exempel en databas). Svaret på denna förfrågan lämnas sedan till servern som i sin tur skickar svaret vidare till webbläsaren. Vid flera förfrågningar startar ett skript för varje förfrågan, och varje skript strävar efter att kunna få tillgång till serverns

resurser för att kunna exekvera, vilket försämrar serverns prestanda och leder till att användaren måste vänta längre innan denne får svaret.

Vid behandling av säkerheten i systemet kräver CGI användaridentifiering i form av användarnamn och lösenord. Dessutom lagrar det IP-adressen till besökare som släpps in för att komma ihåg denna vid senare tillfälle. Detta kan CGI hantera såvida det inte handlar om stora mängder av besökare. Vid flera besök samtidigt blir det lätt för kunniga användare (attackerare) att ’lura’ CGI genom att exempelvis låta det inbilla attackerarens IP-adress som om det hade varit någon annans.

En säkerhetsåtgärd för CGI är att placera alla CGI-program i en katalog direkt efter roten på servern, som bara blir tillgänglig för systemadministratören. Detta kan hindra främmande från att exekvera sina egna CGI-skript på servern och sedan få ut den informationen som de önskar. Men detta är inte fullt säkert eftersom det finns direktiv som tillåter programmerare att kunna exekvera sina egna CGI-skript utanför rot-katalogen (genom att använda sig av specifika filändelser .pl, .cgi, med mera) och sedan förstöra i systemet (till exempel genom att skriva ut innehållet av en viktig fil). CGI-programmet måste vara väl genomtänkt och strukturerat för att kunna hålla säkerheten, och detta beror helt på programmeraren som skriver programmet.

Vid insamling av information om CGI märktes det att de flesta aspekterna som påpekar säkerheten och effektiviteten är lika i olika böcker, men skrivna på olika former. Därför har samma uppfattning om dessa aspekter bildats redan efter läsning av första boken om CGI.

5.2 Active Server Pages (ASP)

Som nämndes i kapitel 2 är Active Server Pages (ASP) en programmeringsmodell som tillåter att dynamiska och interaktiva webbsidor skapas på webbservern, oberoende av användarens webbläsare, eller vilket språk användarens dator stödjer. ASP introducerades av Microsoft med IIS (Internet Information Server 3.0) och tillåter användandet av ett antal olika skriptspråk inom ett enda ASP-skript. De främsta skriptspråken som används i ASP är VBScript och JScript.

Alla ASP filer slutar med .asp, och sådana filer kan innehålla en kombination av text, html-taggar som avgränsas med speciella tecknen (< och >) samt skript-kommandon och output uttryck som avgränsas med tecknen (% och %) (Connolly, m.fl. 1999).

Rent praktiskt fungerar ASP som en sorts HTML-generator där allt innanför avgränsarna processas och eventuellt omvandlas till HTML-kod, och allt utanför skickas till webbläsaren som det är (Ek, m.fl. 1999).

5.2.1 Klientskript och Serverskript

ASP är ett sorts HTML-dokument med skript. Skripten bäddas in i själva HTML-koden för att slippa hålla reda på en massa filer, och för att det då går att skriva den största delen av HTML-koden precis som den skall vara, och sedan lägga in skriptet där det skall producera sina utdata. Skripten kan antigen köras på klienten (webbläsaren) eller på servern. Genom att köra ASP på klienten avlastas servern något vilket är värdefullt vid stora webbplatser där flera användare besöker platsen samtidigt. Men alla klienter har inte stöd för skript, och framför allt inte för olika typer av skript, och detta leder till att informationen inte blir tillgänglig för alla användare. Genom att köra skript på servern fungerar det på vilken klient som helst. En process som körs på servern kan komma åt serverns resurser, men nackdelen med

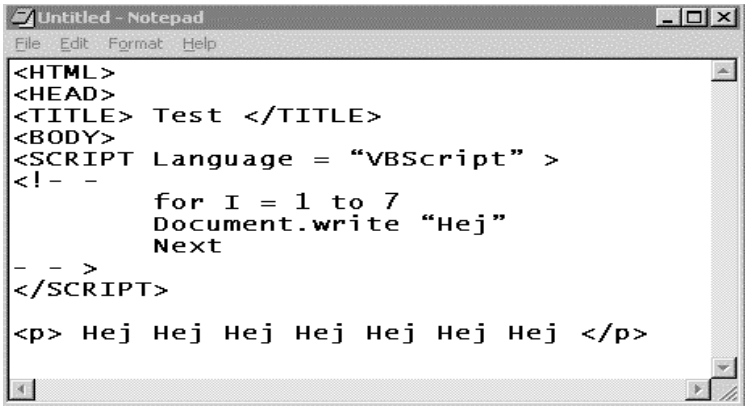
att köra skript på servern är att den kan bli tungt belastad vid hantering av stora mängder av samtidiga besök (Ek, m.fl. 1999).

Skillnaden mellan klientskript och serverskript är liten. Det är egentligen inte skriptspråken som skiljer sig åt från klient till server, utan reglerna för vad som skall skrivas och var det skall skrivas som skiljer. Skrivs ett skript för servern kommer programmeraren åt serverns objekt, som till exempel objekten som ingår i ASP. Ett skript på klienten kommer däremot åt klientens objekt, som kan vara listrutor, textrutor och inbyggda objekt i webbläsaren. Ett skript på servern har inte rätt att visa dialogrutor, som till exempel felmeddelande, med mera (Ek, m.fl. 1999).

Serverskript och klientskript körs vid olika tillfällen. När en webbläsare kontaktar en webbserver med en förfrågan om ett ASP-dokument, inleds arbetet med att servern skickar ASP-dokumentet till ASP-tolken som i sin tur processar all ASP-kod och alla ”uttryck” översätts till HTML-kod. Därefter skickas sidan till klienten (webbläsaren) som översätter HTML-koden till läsaren, och när dokumentet hamnar hos läsaren börjar eventuella klientskript fungera.

Exempel på ett vanligt skript som skickas från servern så att det bearbetas av klienten:

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE> Test </TITLE>
 4 <BODY>
 5 <SCRIPT Language = “VBScript” >
 6 <!- -
 7 for I = 1 to 7
 8 Document.write “Hej”
 9 Next
10 - - >
11 </SCRIPT>
12 <!- -
13 <% for I = 1 to 7 %>
14 Hej
15 <% NEXT %>
16 - - >
17 </BODY>
18 </HTML>
```



Figur 8: Resultat av ASP sidan i webbläsaren. Fritt från Ek, m.fl. 1999.

Raderna 13-15 bearbetas av ASP på servern eftersom de avgränsas med tecknen (% och %). Allt som återfinns mellan dessa tecken bearbetas på servern och skickas till

klienten som HTML-kod, oavsett var i dokumentet det står. Resultatet av exekvering av dessa skript kan se ut som i figur 8, när de hamnar hos klienten.

Nackdelen med att bädda in ASP-koden i HTML-sidorna är att det blir svårare att felsöka skriptet, och det blir också lättare att (både avsiktligt och oavsiktligt) förstöra skriptet vid ändringar på webbsidan.

5.2.2 ASP och Webbservern

ASP-skript körs multitrådigt, vilket betyder att det bara startas en upplaga av skriptet på servern, istället för en upplaga per besökare som använder samma skript. Användaren ser aldrig server skriptet, för när det når klienten är det HTML-kod (Ek, m.fl. 1999).

När ASP kompilerar koden som hittas i en sida lagras exekveringen i ett cache-minne så att koden inte behöver kompileras om igen, såvida källan till denna sida inte ändras. Detta resulterar i att nya förfrågningar till sidan behandlas snabbare, och klienten behöver inte vänta länge innan den får svaret från servern (Anderson, Denault, Francis, Gibbs, Gregorini, Homer, McQueen, Schenken, Robinson & Williams, 2000).

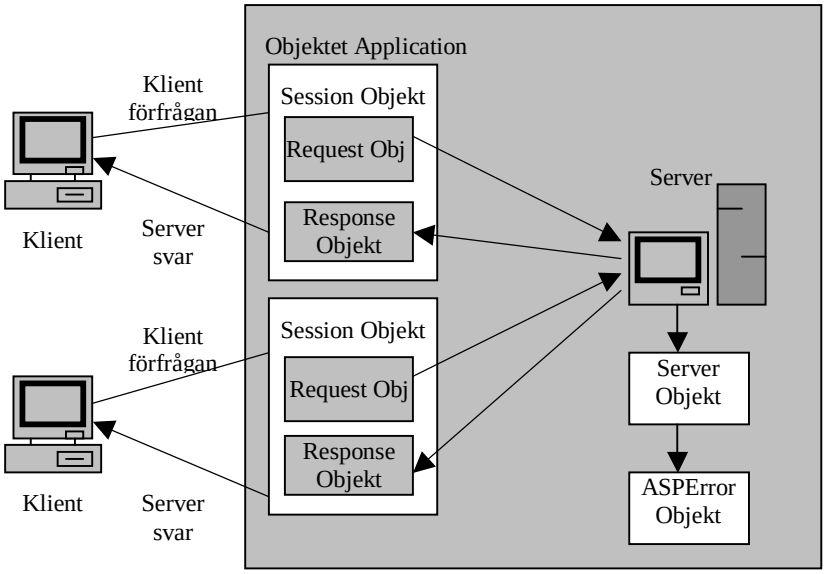
Förfrågningar och svar mellan webbläsaren och webbservern hanteras med ett antal så kallade ASP-funktioner och inbyggda objekt som finns i ett funktionsbibliotek. Funktionsbiblioteket läses in av servern så att serverskriptet använder sig av detta (Ek, m.fl. 1999).

Anderson m.fl. (2000) nämner och beskriver dessa objekt enligt nedan:

Request
Response
Application
ASPError
Server
Session

Request och *Response* är de två huvudobjekten som mappar direkt till förfrågan och svaret, som http-protokollet använder för att överföra information mellan klienten och webbservern. De andra objekten erbjuder ett antal funktioner som utnyttjas för skriptning av ASP-applikationen. Nedan ges en kort beskrivning av dessa objekt:

Objektet Application (se figur 9) skapas då första förfrågan ställs från klienten till webbservern. Det tillhandahåller en plats för lagring av referenser om variabler och objekt för alla besökare som besöker ASP- sidan inom en bestämd period. *ASPError* objektet tillhandahåller information om det senast inträffade felet i ASP. *Serverobjektet* erbjuder metoder och egenskaper som är bra vid skripting med ASP, och metoder för att översätta strängar till rätt format så att de kan användas i URL:er eller HTML. Till varje besökare som begär en ASP-sida skapas ett *Sessionsobjekt* som tillhandahåller plats för referenser om variabler och objekt till sidorna som besökts av besökaren under sessionens livstid.



Figur 9: Relationen mellan ASP och processen att skapa ASP-sidor, fritt från Anderson, m.fl. 2000.

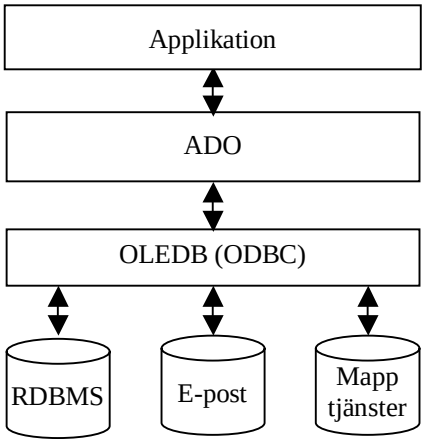
ASP tillhandahåller flexibilitet vilket gör att den körs utan prestandakrångligheter för webbservern. ASP är multitrådig och optimerad för att hantera stora mängder av användare samtidigt. ASP-skript (som till exempel VBScript och JScript) hos klienten kan hantera vissa typer av datainmatning innan det skickas till servern. Ett exempel på detta; en användare fyller i ett formulär som finns på en webbsida och som sedan skall skickas över nätet till webbservern. När användaren fyller i formuläret och klickar på knappen för att skicka formuläret kontrolleras innehållet av skriptet hos klienten. Om skriptet upptäcker något fel i inmatningen, eller om något fält lämnats tomt, får användaren rätta till detta, och sedan när allt blir klart skickas formuläret till servern. Det här sättet att hantera data leder bland annat till tidsvinst, minskad trafik över nätet och avlastning av servern (Ek, m.fl. 1999).

En ytterligare egenskap hos ASP är att den kan upptäcka blockeringssituationer som inträffar när en förfrågan blockeras av externa resurser medan den exekveras. I det fallet skapar ASP automatiskt flera trådar så att ytterligare förfrågningar körs parallellt med den exekverande förfrågan. Vid överbelastning av processorn ser ASP till att minska antalet tillgängliga trådar för att reducera trådomväxlingen som inträffar när stora mängder av ej blockerade förfrågningar körs samtidigt (Anderson, m.fl. 2000).

5.2.3 ASP och Databaser

Ett av de främsta användningsområdena för ASP är att använda det som frågespråk till en databas. För att det skall fungera måste databasen konfigureras så att det går att kommunicera med servern och ställa frågor till den, och det som krävs först och främst är att koppla databasen via en ODBC (Open DataBase Connectivity) (Ek, m.fl. 1999).

Utöver ODBC krävs det att instansiera ett databasobjekt och arbeta mot detta. Ett sådant objekt kallas för ADO (ActiveX Data Object) som är speciellt utvecklat för att vara hanterbart på Internet (se figur 10). ADO är ett litet och behändigt objekt som är



Figur 10: ADO teknologi, fritt från Anderson, m.fl. 2000.

kontaktytan mellan ASP-sidan och databasen där den information som presenteras finns lagrad (Ek, m.fl. 1999).

Enligt Anderson m.fl. (2000) innehåller ADO ett antal egna objekt, och de viktigaste objekten är; Connection, Commando, Record, Stream och Recordset som är det viktigaste objektet.

På grund av viktigheten hos ADO-objektet för hantering av information mellan ASP och databaser ges en beskrivning av de ovan nämnda objekten.

Objektet Connection är kommunikationslänken mellan webbapplikationen och databasen via en datakälla (till exempel en ODBC-källa). Objektet innehåller ett antal egenskaper och metoder, egenskaperna är exempelvis att koppla upp eller stänga objektet till datakällan, starta och avsluta en transaktion, med mera. Metoderna i sig består av några egna objekt, och dessa objekt innehåller bland annat informationen som använts för att skapa kopplingen till datakällan, för att isolera transaktioner på olika nivåer, ange behörighetstypen, med mera (Ek, m.fl. 1999).

Connection objektet används också för att konfigurera uppkopplingen till databasen och ange en standarddatabas för kopplingen. Det kan också utnyttjas för att bestämma vilken isolationsnivå som ska användas för transaktioner, skapa och stänga fysisk koppling med databasen och hantera fel som uppstått i databashanteringen.

Objektet Command är det andra kommandot i ADO-hierarkin och syftet med objektet är att optimera utförandet av SQL-frågor och anropa i databasen sparade frågor. Objektet har egenskaper och metoder som tar hand om kommandon som skall utföras, exekvering av SQL-uttryck, med mera.

Objektet tillåter programmeraren att definiera det exakta värdet (datatyp och längd) för parametrarna som används, och använda till exempel returvärde och output-parametrar som tar emot värde från kommandon (Anderson m.fl. 2000).

Command kan också utnyttjas för att köra lagrade frågor, men den stora nyttan av objektet är dess hantering av parametrar. Genom att använda objektet och dess parametrar, effektiviseras serverns prestanda. Ett sätt för att undersöka detta kan visas

med två exempel, det första exemplet använder inget Command objekt medan det andra gör.

```
<!-- #Include file=”ADOVBS.INC” -->
Set ConMitt = Server.CreateObjekt(“ADODB.Connection”)
ConMitt.open “Suppo”
Set RstMedlem = Server.CreateObjekt(“ADODB.Recordset”)
RstMedlem.Open “Medlemsreg”, ConMitt, adOpenDynamic,
adLockPessimistic
RstMedlem.AddNew
RstMedlem(“id”) = iReknare
RstMedlem(“Fnamn”)= sFornamn
RstMedlem(“Efternamn”) = sEfternamn
RstMedlem(”Telefon”) = sTelefon
RstMedlem.Update
```

Samma exempel fast med Command-objektet kan se ut som:

```
<!-- #Include file=”ADOVBS.INC” -->

Set ComTest = Server.CreateObjekt(“ADODB.Command”)
ComTest.ActiveConnection= “Drive={Microsoft Access Driver
(*.mdb)};DBQ=F:+UTV\HexDB\support2.mdb”

ComTest.CommandText = “Insert into Medlemsreg( id, Förnamn, Efternamn, Telefon)
Values(?,?,?,?)”

Set prmNy = ComTest.CreateParameter(,adInteger,,,iReknare)
ComTest.Parameters.Append prmNy
ComTest.CreateParameter(,adVarChar,,Len(sFornamn),sFornamn)
ComTest.Parameters.Append prmNy
ComTest.CreateParameter(,adVarChar,,Len(sEfternamn),sEfternamn)
ComTest.Parameters.Append prmNy
ComTest.CreateParameter(,adVarChar,,Len(sTelefon),sTelefon)
ComTest.Parameters.Append prmNy
ComTest.Execute Recordsaffected,,adCmdText
```

I det första exemplet skapas en postsamling och det med en Cursor[4]. Därefter sker skapandet av den nya posten i databasen. Det är stort slöseri med datorns prestanda för att mata in en post eftersom det kräver att hela förfarandet gås igenom från början till slut med exekvering för varje rad istället för att skicka hela posten en enda gång. Om tabellen dessutom innehåller stora mängder data tar det även lång tid att föra det över datanätet.

I det andra exemplet skapas Command-objektet och egenskapen CommandText fylls med en halvfärdig SQL-sats, som slutar med ett antal frågetecken som är lika med antalet fält i den nya posten. Därefter skapas Parameterobjektet och fylls med värden. Slutligen sker det enda anropet till databasen genom att metoden Execute utförs på Command objektet.

Ett annat alternativ istället för att använda parameter-objektet är att bygga en egen SQL-sträng, som är ännu effektivare, beroende på vilka fördefinierade konstanter som används, som antingen kan göras med hjälp av VBScript eller med JavaScript (Ek m.fl. 1999).

Objektet Recordset är det viktigaste och mest använda objektet i ADO-hierarkin eftersom detta objekt innehåller all data som finns och resultatet av en samling

[4] Cursor tillhandahåller ett sätt för att hantera poster inom Recordset-objektet, och finns i fyra typer; statisk, dynamisk, a forward-only, och keyset. För mer information om Cursor se (Ek, m.fl. 1999).

förfrågan i form av en samlig av poster. Metoderna i objektet gör det möjligt att bland annat uppdatera innehållet i postsamlingen, sortera posterna i en viss ordning, söka igenom posterna och spara alla uppdateringar till disk. Egenskaperna hos objektet behandlar tillståndet hos den aktuella posten, vilken typ av Cursor som gäller för den aktuella postsamlingen och filtrerar postsamlingen (Ek, m.fl. 1999).

Objektet Record arbetar gemensamt med Recordset objektet och underlättar tillgängligheten av begärd data i databasen.

Objektet Stream innehåller en binär eller textbaserad dataström som underlättar tillträdet till innehållet i en fil. Objektet har direkt samarbete med Record objektet.

5.2.4 Försiktighetsaspekter

En applikation kan i ASP bestå av en eller flera sidor, där data delas mellan sidorna och användare. Applikationen innehåller händelser som kan aktiveras, och instansen av ett objekt kan delas av sidorna. Applikationen kan köras i ett separat minnesutrymme för att försäkra att vid en krasch inte förstöra något annat. Körningen av en applikation kan stoppas utan att andra applikationer påverkas (Ek, m.fl. 1999).

En ASP applikation innehåller ofta känslig data, därför är säkerheten hos databasen som applikationen kopplas till en viktig åtgärd. Eftersom SQL-servern är den populäraste databasen för ASP applikationer skall aspekterna som berör säkerheten i denna diskuteras i detta avsnitt.

En systemadministratör kan konfigurera SQL-servern med avseende på säkerhet efter dennes behov. Administratören kan skapa nya användare till databasen och ge de rättigheter som bedöms vara nödvändiga, eller skydda servern från obehöriga, med mera (Anderson m.fl. 2000).

Varje databas på servern har sina egna regler och dessa regler kan ges till varje användare enskilt beroende på vad denne har för rättigheter, eller till en grupp av användare som har samma rättigheter. Därför kan säkerheten i databasen betraktas i form av olika nivåer, och utifrån dessa nivåer delas rättigheterna ut till databasens användare. Reglerna för en databas kan bara användas inom den berörda databasen. En sådan struktur ger en säker miljö till länkarna mellan ASP och databasen (Anderson m.fl. 2000).

När IIS och ASP installeras i ett operativsystem som exempelvis Windows 2000, skapas en default ASP webbapplikation som innehåller en uppsättning egenskaper som bestämmer hur uppträdandet sker i förhållande till IIS. Det finns en fil som kallas för global.asa, som hanterar uppförandet av den default skapade applikationen. Filen tillåter användandet av globala variabler, och även användandet av metoderna i applikations och sessions objekten. Denna fil lagras i en rot katalog och kan inte läsas, den hjälper också till med att skapa händelsehanterare för webbsidan eller applikationer. Exempel på händelsehanterare är Session_OnStart, Session_OnEnd, Application_OnStart och Application_OnEnd. Dessa körs första gången när någon användare kommer in i webbsidan och avslutas när denne lämnar sidan. Men hur kan denna fil erbjuda säkerhet? Säkerheten fås i första hand genom att användare inte kan begära filen genom sina webbläsare från webbservern. Detta kan utnyttjas genom att spara databaskopplingssträngen som innehåller användarnamn och lösenord i global.asa filen och sedan referera till den från applikationen. När någon användare sedan begär global.asa filen från servern, kommer följande felmeddelande att dyka upp (Anderson m.fl. 2000):

HTTP/1.1 Request for GLOBAL.ASA Not Allowed

Det är en säker åtgärd att spara kopplingssträngen till databasen i global.asa filen jämfört med att överföra användarnamn och lösenord och andra kopplingsdetaljer i en ASP fil, webbsida, eller webbapplikation över nätet. Detta hindrar surfaren från att ta reda på koden som processas för tillfället av webbservern (Anderson m.fl. 2000).

5.2.5 Säkerhetsåtgärder

Den främsta säkerhetsåtgärden som kan utgås från för säkerheten i ASP är att konfigurera webbservern genom att till exempel lägga alla filer av samma typ i ett bibliotek, vilket underlättar skapandet av olika rättigheter till olika bibliotek. Vid installationen av exempelvis Windows 2000 bör vissa komponenter väljas ut så att de inte kommer med i installationen. Exempel på sådana komponenter är (Anderson, m.fl. 2000):

Front Page Server Extensions

Remote Data Services.

”Front Page Server extensions” kan tillåta externa användare att lägga egna data i webbservern och förstöra i systemet. Tänk att någon användare har tillgång till en FTP server för en webbsida benägen på webbservern och skickar följande kod:

```
ftp hos.com
mkdir tro
cd tro
put trojanfile.exe
exit
```

Om denne sedan skriver http://xxxx.com/tro/trojanfile.exe på dess webbläsare kommer trojan filen att köras på servern.

Det finns flera skäl för att inte välja att installera ”Remote Data Service” på Internet Information Server. Ett av dessa skälen är att användare kan skapa egna formulär på sina egna webbservrar och sedan försöka att koppla sig till databasen via ODBC (Anderson, m.fl. 2000).

5.2.6 Sammanfattning

I detta kapitel har CGI och ASP behandlats var för sig, och avsikten med detta är att öka förståelsen för hur dessa fungerar för att underlätta utvärderingen av dess säkerhet och effektivitet. Men innan övergången till nästa kapitel kan det vara av betydelse att gå igenom en sammanfattning för det som har diskuterats om ASP hittills:

ASP är en programmeringsmodell som tillåter skapandet av dynamiska och interaktiva webbsidor på webbservern oberoende av webbläsaren hos användaren, eller vilket språk användarens dator stödjer. ASP tillåter användandet av olika skriptspråk i ett enda ASP-skript och skriptspråken som främst används i ASP är VBScript och JScript.

Skript kan antingen köras på klienten eller på servern. Genom att köra skript på klienten avlastas servern, men det är inte alltid ett bra alternativ eftersom alla klienter inte har stöd för skript. Däremot genom att köra skript på servern kan alla klienter ta del av informationen, och nackdelen med detta är att det belastar servern något. ASP körs multitrådigt vilket innebär att det bara exekverar ett skript för en eller flera förfrågningar.

När webbservern får förfrågan om en .asp fil, kallar den ASP (asp.dll filen) som går igenom sidan och exekverar alla kommandon som ligger mellan tecknen (% och %) sedan skickar servern vidare sidan till webbläsaren i form av HTML-kod som sedan tolkas av klienten.

När ASP exekverar (eller kompilerar) en .asp fil sparar det innehållet i ett cache minne så att sidan inte behöver kompileras om igen såvida innehållet i sidan inte ändrats. ASP behandlar förfrågningar genom att utnyttja ett antal funktioner och objekt som finns lagrade i ett funktionsbibliotek som läses av servern. Dessa funktioner och objekt gör att ASP blir flexibel och inte orsaka prestandakrångligheter med servern. För att komma åt den begärda informationen från databasen krävs det en ODBC-källa och ett databasobjekt (ADO). ADO innehåller ett antal objekt som utnyttjas för koppling och hantering av information mellan ASP och databasen. Varje objekt i ADO har sina egna egenskaper och metoder som används för hantering av bland annat kommunikation, säkerhet och prestanda.

ASP kan upptäcka blockning och ser till att resurser inte blockerar varandra, genom att skapa nya trådar för varje ny förfrågan, och när den märker att servern börjar ’klaga’ minskar ASP skapandet av dessa trådar.

Säkerhetsaspekter kan tas genom att systemadministratören konfigurera databas-servern sådan att bibliotek och användare får olika rättighetsnivåer, och genom att använda filen global.asa som bara är tillgänglig för systemadministratören. För att hålla säkerheten kan en del åtgärder tas vid installationen av servern, för att förhindra främmande från att attackera och förstöra systemet.

I kommande kapitel analyseras de aspekter som påpekar säkerhet och effektivitet hos CGI och ASP och utifrån det görs en utvärdering av säkerheten och effektiviteten för teknologierna i form av en jämförelse mellan motsvarande aspekter hos varje teknologi.

6 Analys

I detta kapitel analyseras CGI och ASP med avseende på effektivitet och säkerhet utifrån diskussionen som presenterades i kapitel 5. Avsikten med analysen är att visa hur de behandlade teknologierna (CGI och ASP) hanterar säkerhet och effektivitet på ett närmare sätt. Analysen görs enligt följande; för varje aspekt som påverkar säkerheten eller effektiviteten positivt, ges teknologin en vit boll, och för varje svaghet (aspekt som påverkar säkerheten eller effektiviteten negativt) hos teknologin ges den en svart boll. Kapitlet avslutas med en tabell som kombinerar de erhållna resultaten.

6.1 Säkerhet hos CGI och ASP

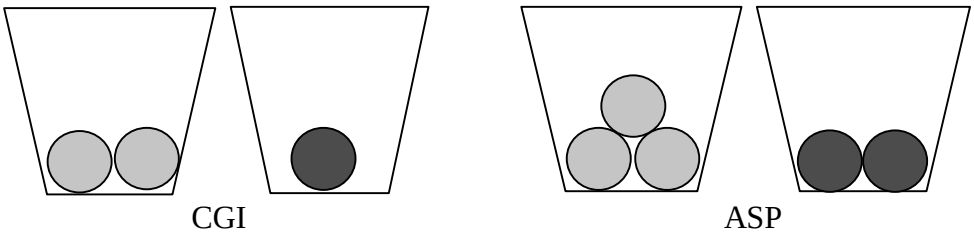
Som en säkerhetsåtgärd kan CGI-programmet placeras i en rotkatalog på webbservern så att systemadministratören får direkt kontroll över katalogen, vilket hindrar andra från att exekvera sina egna skript på servern för att förstöra, eller få ut information efter dess önskemål. Detta kan ASP hantera genom att välj bort vissa komponenter (se kapitel 5.2.5) vid serverinstallation för att hindra främmande (obehöriga) från att attackera servern och förstöra, eller få ut information. Därför får CGI och ASP en vit boll var. Men det finns direktiv som tillåter programmerare som vill komma åt information på en server att exekvera sina CGI-program utanför rotkatalogen, och sedan komma åt den information som de behöver. Med ASP kan motsvarande problem inträffa om en del komponenter (se kapitel 5.2.5) inte väljs bort vid serverinstallation, vilket innebär att de tidigare erhållna vita bollarna ersätts med svarta bollar till båda teknologierna.

Vid besök från användare går det att kontrollera användarens tillträde till informationskällan genom att kontrollera dennes IP-adress som finns lagrad i CGI:s REMOTE_HOST miljövariabel, vilket ger CGI en vit boll. Men REMOTE_HOST kan snart falla på olika sätt genom att vid stort antal besökare kommer variabeln inte att kunna skilja mellan användarens IP-adresser, eller genom att kunniga användare låtsas ha en annan IP-adress än vad de egentligen har, och med detta förlorar CGI den vita bollen som den fick tidigare. I ASP kan information om användare (enligt beskrivningen i kapitel 5.2.4) sparas i filen global.asa som finns lagrad på roten på webbservern. Filen kan inte läsas och kan ej heller nås från användare. Eftersom REMOTE_HOST variabeln i CGI kan luras, men inte filen global.asa hos ASP, får ASP en vit boll.

När det gäller strukturen av koden i CGI och ASP så innehåller CGI-program bara dess egen kod och kan inte blandas med HTML-kod som fallet är hos ASP där skript bäddas in i HTML-sidor. Genom att bädda in ASP-koden i HTML-sidor blir det svårare att felsöka skriptet, och lättare att förstöra skriptet vid ändringar på webbsidan. Vilket ger CGI en vit boll och ASP en svart boll.

En av de bra aspekterna som finns hos ASP är att ASP kan upptäcka blockningssituationer som inträffar när en förfrågan blockeras av externa resurser under tiden exekvering pågår. Eftersom en sådan aspekt saknas hos CGI, får ASP en vit boll.

Med detta får CGI och ASP följande resultat om säkerhet:



Analysering av säkerhet mellan CGI och ASP

6.2 Effektivitet hos CGI och ASP

För varje förfrågan som webbservern får från klienten och lämnas till CGI, och för varje svar som CGI får från databasen, måste webbservern konvertera data från och till HTML format i form av HTML-dokument. Motsvarande situation återfinns också hos ASP, då varje förfrågan från klienten resulterar i att hela den begärda .asp filen måste gås igenom för att exekvera eventuella serverskript. Här hanteras förfrågningar på motsvarande sätt hos CGI och ASP, varför de får en svart boll var.

På webbservrar där CGI finns kan en aktiv server ha flera skript som körs samtidigt, och vart och ett körs med sin egen miljö. Däremot är det så att varje skript strävar efter serverns resurser (minne, processorkraft, med mera) vilket kan minska serverns prestanda. ASP hanterar detta på ett annorlunda sätt, där ASP-skript körs multitrådigt, vilket innebär att det bara startar en upplaga på servern för en eller flera förfrågningar. ASP skapar automatiskt flera trådar så att ytterligare förfrågningar körs parallellt med en exekverande förfrågan. Vid överbelastning av servern slutar ASP med att skapa nya trådar för att reducera trådomväxlingen som inträffar när stora mängder av förfrågningar körs samtidigt. Därför får CGI en svart boll och ASP får en vit boll.

När CGI tar reda på svaret för en förfrågan skickar det svaret tillsammans med en header som innehåller information om svaret. Header kan antingen skickas som en del-header eller som en hel-header. Är det hel-header skickar CGI svaret direkt till klienten utan inblandning av webbservern. Det finns motsvarande aspekt hos ASP, efter att en .asp fil har kompilerat färdigt, lägger ASP kompileringen i ett cache-minne så att det blir snabbare att använda detta vid ytterligare förfrågningar om samma sida (såvida sidan inte ändras). Dessa aspekter ger var och en av CGI och ASP en vit boll var.

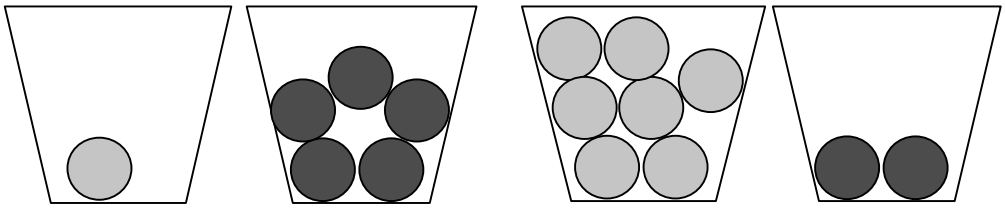
CGI-skript kan endast köras på servern vilket belastar servern vid många samtidiga förfrågningar. ASP-skript kan antingen köras hos klienten eller på servern, och genom att köra ASP-skript hos klienten avlastas servern något. Här är CGI och ASP lika när skript körs på servern, men däremot är ASP bättre genom möjligheten att köra skript hos klienten. Detta ger CGI och ASP en svart boll var, och dessutom får ASP en vit boll för möjligheten att köra skript hos klienten.

För varje förfrågan till webbservern måste servern generera en ny process eller tråd för varje CGI-skript, detta för att skripten körs som separata program på servern. ASP körs multitrådigt och det startas bara en upplaga av skriptet på servern vare sig om det är en eller flera förfrågningar. Detta ger CGI en svart boll och berättigar ASP till en vit boll.

När någon användare fyller i ett formulär och sedan skickar det till servern, är det servern och inte CGI som vägrar att ta emot formuläret ifall användaren matat in fel värden, eller glömt fylla i något fält. ASP-skript som körs hos klienten kan hantera vissa typer av data inmatningar innan de skickas till servern, vilket minskar behandlingstiden för formuläret och även minskar trafiken på nätet och arbetet på servern. Med detta får CGI en svart boll och ASP får en vit boll.

ASP behandlar förfrågningar genom att utnyttja ett antal funktioner och objekt som finns lagrade i ett funktionsbibliotek som läses av servern. Dessa funktioner och objekt samarbetar med ett databasobjekt (ADO) för att kunna skicka och hämta information till och från destinationen (databasen). Det strukturerade samarbetet mellan objekten i funktionsbibliotek och objekten i ADO underlättar serverns arbete och ökar dess prestanda. Detta förfarande saknas hos CGI, vilket gör att ASP får en vit boll.

Utifrån denna analys om effektivitet hos CGI och ASP erhålls följande resultat:

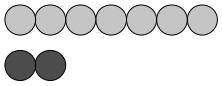


CGI ASP

Analysering av effektivitet mellan CGI och ASP

6.3 Resultat

Utifrån de erhållna resultaten för säkerheten och effektiviteten hos CGI och ASP skapas en gemensam tabell som kombinerar dessa resultat, för att lättare kunna belysa skillnaderna mellan teknologierna. Sammanfattningsvis har CGI erhållit två vita bollar och en svart boll för säkerhet, medan ASP har samlat tre vita bollar och två svarta bollar för detta. CGI har dessutom fått en vit boll och fem svarta bollar för effektivitet, medan ASP har erhållit sju vita bollar och två svarta bollar respektive, vilket illustreras i tabell 1:

	Säkerhet	**Effektivitet**
CGI		
ASP		

Tabell 1: kombination av säkerhet och effektivitet hos CGI och ASP. Den ljusa färgen påpekar de starka sidorna hos teknologin, medan den mörka färgen påpekar de svaga sidorna.

I nästa kapitel behandlas slutsatsen utifrån denna analys, där bestäms vilken av teknologierna CGI eller ASP som har bättre säkerhet och effektivitet.

7 Slutsats

Enligt analysen i kapitel 6 har både CGI och ASP sina svaga och starka aspekter som påverkar säkerheten och effektiviteten hos de båda. I detta kapitel kommer dessa aspekter att användas för att bedöma vilken av dessa två teknologier som har bättre säkerhet och effektivitet.

Genom att studera tabell 1, som kombinerar analysen av teknologierna, erhålls en direkt inblick om det slutgiltiga resultatet, eftersom det finns en skillnad mellan CGI och ASP både när det gäller säkerhet och effektivitet. Skillnaden med avseende på säkerhet är tydligen inte så stor mellan teknologierna, men den stora skillnaden beaktas vid effektivitet. Enligt kapitel 6.1, har både CGI och ASP sina egna metoder för att skydda data, men ASP övervinner CGI i detta sammanhang med bland annat den främsta anledningen, att identifikationen av besökare sker med hjälp av global.asa filen, som är omöjlig att komma åt för en besökare. Trotts att ASP har en svart boll mer än vad CGI har, bedöms ASP hantera säkerheten bättre än CGI med tanke på antalet vita bollar och säkerheten som uppnås med hjälp av filen global.asa. Med denna slutsats stärks den första delen i *förväntat resultat* som redovisades i kapitel 3.3, och som hävdade att ASP hanterar säkerheten bättre än CGI.

Effektiviteten hos CGI är betydligt svagare jämfört med den hos ASP. CGI har bara fått en vit boll för detta, men vid jämförelse med ASP har ASP orkat samla sju vita bollar. Även antalet svarta bollar hos CGI är mer än det hos ASP, och med detta visas den andra delen av påståendet i *förväntat resultat* i kapitel 3.3, som säger att ASP är bättre än CGI även när det gäller effektivitet.

Slutsatsen av de ovan erhållna resultaten visar på att ASP är bättre än CGI både med avseende på säkerhet och också effektivitet. Detta ger lösningen till problemet som presenterades i kapitel 3, och fokuseringen i kapitel 3.2. Slutsatsen kommer bland annat att underlätta för personer eller företag som vill koppla sina databaser till Internet (se kapitel 3.1).

7.1 Diskussion

Avsikten med arbetet i denna rapport var att utvärdera teknologierna CGI och ASP, och sedan komma fram till en slutsats som visar vilken av dessa teknologier som är bättre med avseende på säkerhet och effektivitet.

Arbetet inleddes med att studera CGI och ASP och undersöka vilka andra tillgängliga tillvägagångssätt som kan användas för samma ändamål. I början av arbetet saknades det kunskaper om CGI och ASP vilket satte press för intensiv start, där samlades information om dessa, som bildade en grund att utgå ifrån. Det visade sig att ett sådant, eller liknande arbete inte har genomförts innan, men däremot finns det tillräckligt med böcker som behandlar CGI och ASP i separata sammanhang (se kapitel 4.7.1). Därför valdes litteraturstudie som metod för utförandet av arbetet. För att underlätta förståelsen för läsaren studerades varje teknologi enskilt i genomförandet. För att kunna bilda en uppfattning om teknologierna måste de studeras var för sig så att information som fås sedan kan diskuteras och jämföras. Detta ansågs vara tydligare än att skriva om båda teknologierna samtidigt, vilket annars kan leda till att läsaren sedan lätt blandar ihop informationen som tillhör CGI eller ASP.

Efter genomförandet behandlades analysen av säkerhets- och effektivitets- aspekter hos CGI och ASP, där varje aspekt hos den ena teknologin jämfördes med motsvarande aspekt hos den andra. För att rättvist kunna bedöma skillnaderna mellan teknologierna valdes det att dela ut en ljus boll för den förbättrade aspekten hos teknologin, och en mörk boll för den svaga aspekten. Vidare motivering till detta sätt att jämföra teknologierna är att de jämförs med avseende på vilken som har bättre säkerhet eller effektivitet, och inte på hur mycket som skiljer säkerheten hos det ena teknologin gentemot den andra. Dvs så fort någon av teknologierna visade någon skillnad skulle den ha fått antingen en vit eller en svart boll.

Jämförelsen utfördes separat för säkerhet och effektivitet, där varje jämförelse tydligt visade hur många bollar varje teknologi fick. Sedan presenterades resultaten i en tabell som illustrerar det slutgiltiga antalet bollar för de båda. Detta blev utgångspunkten till den slutgiltiga slutsatsen som visade att ASP har bättre säkerhet och även påvisade bättre effektivitet än CGI, och den erhållna slutsatsen stämmer väl överens med det förväntade resultatet i kapitel 3.3.

7.2 Fortsatt arbete

I detta arbete har utvärdering av säkerhet och effektivitet hos CGI och ASP behandlats, och slutsatsen för detta visar på att ASP är bättre än CGI både när det gäller säkerhet och effektivitet. I kapitel 5 stöds vissa åtgärder med kod-exempel för att tydligare förklara hur det som beskrivs teoretiskt kan uppnås vid applicering av verkliga åtgärder för att skydda data, eller för att få bättre effektivitet hos den aktuella teknologin.

Förslag på fortsatt arbete är att komplettera arbetet som genomförts i denna rapport, genom att konstruera en applikation som kan användas till att utvärdera både CGI och ASP i praktiska situationer (se bilaga A för länkar till kod-exempel). Applikationen kan exempelvis användas för att först testa effektiviteten hos CGI genom ett antal olika datainmatningar, och sedan köra applikationen på liknande sätt i ASP. Efter mätning av effektiviteten hos dessa kan samma applikation användas för att mäta säkerheten hos dessa teknologier. Resultaten som uppnås kan sedan användas så att de kombineras med det slutgiltiga resultatet som har redovisats i denna rapport.

CGI och ASP kommer förmodligen att utvecklas kontinuerligt, och särskilt ASP. Därför kan ytterligare ett förslag på fortsatt arbete vara att utvärdera säkerheten och effektiviteten igen, dock anses detta inte nödvändig innan en tydlig förändring av dessa teknologier har inträffat.

7.3 Framtiden för CGI

CGI är en teknologi som används som en gateway mellan databaser och Internet. Sedan den introducerades har CGI fått förtroendet från företag och personer som ansåg sig i behov av en sådan teknologi. Trotts att Microsoft har introducerat ASP som en variant och ersättare för CGI, används CGI fortfarande, och en av orsakerna för detta är oberoendet av CGI som gör att den kan användas på webbservrar oavsett typen av operativsystem som finns installerat på servern. Dessutom, är möjligheten att kunna skriva CGI-program med de flesta programmeringsspråken (Perl, C, C^{++}, med mera) en annan anledning som lett till den stora spridningen av CGI.

CGI kommer förmodligen att finnas kvar under de kommande åren, dock med en viss förbättring av dess hantering av säkerhet, helt enkelt därför att säkerheten i CGI är helt beroende av programmeraren som skapar programmet. Därför finns det möjlighet

för CGI-programmerare att betrakta detta som en självutmaning, och skapa skript (program) som klarar av att hantera de flesta attackeringstyper som siktar på att komma åt skyddad data via CGI. Effektiviteten hos CGI kommer alltid att vara begränsad, och det på grund av den viktiga anledningen att servern måste exekvera en ny process för varje CGI-skript. Den höga kompatibiliteten mellan CGI och webbservrar tyder också på fortsatt användning av CGI.

7.4 Framtiden för ASP

ASP är en annan teknologi som utvecklades av Microsoft för att kunna bädda in enklare programmeringskod (ASP-kod) i HTML-koden. Avsikten med detta är att kunna skapa dynamiska webbsidor (vilket även är målet för CGI), och kunna komma åt information från program på webbservern. ASP har på en kort tid blivit populär bland webbprogrammerare, och särskilt bland de som föredrar Visual Basic eller Java. När ASP introducerades kunde den bara fungera med Microsofts produkter (Windows NT/95 och senare) men idag finns det program som tillåter användandet av ASP även på UNIX servrar. Detta är en väsentlig punkt att utgå ifrån, att ASP kommer att fortsätta sin spridning i framtiden, inte bara i Windows-miljö utan även på andra plattformar. Det kommer dessutom nya versioner av ASP samtidigt som det ’produceras’ nya böcker som utförligt behandlar varje version, vilket attraherar användandet av ASP jämfört med andra liknande teknologier. En annan orsak för att ASP bedöms fortsätta sin spridning inom de kommande åren är att användare (företag, privata personer) alltid vill följa med det nyaste inom utvecklingen. Sålänge ASP utvecklas ständigt i form av nya versioner, kommer det att gynnas av användarnas nyfikenhet på det nya. Här bör även tas hänsyn till tillgängligheten av IIS, jämfört med exempelvis kostnader för att skaffa en Unix maskin. Dessa orsaker, tillsammans med den förbättrade säkerheten och särskilt effektiviteten hos ASP, leder till bedömningen att ASP kommer att hävda sig mer och mer i framtiden.

7.5 CGI eller ASP?

CGI och ASP är inte riktigt samma sak, men de är två teknologier som används till nästan samma ändamål, och har många likheter sig emellan. Trotts likheterna mellan dessa teknologier finns det skillnader när det gäller hur dessa hanterar säkerhet och effektivitet. Enligt arbetet i denna rapport har ASP bättre förutsättningar för att hantera säkerhet jämfört med CGI, och även mycket bättre effektivitet. Vid valet av en sådan teknologi, kan det vara lätt för gamla programmerare att välja CGI med tanken på deras erfarenhet med att programmera i exempelvis programmeringsspråk som Perl. Det har dock visat sig att Perl även kan användas i ASP (Ronne, 2000), som en av många andra språk (VBScript, JScript, med mera). Utifrån denna avslutande diskussion och slutsatserna som gjordes tidigare, lämnas valet till programmeraren.

Referenslista

Anderson, R. Denault, D. Francis, B. Gibbs, M, Gregorini, M. Homer, A. McQueen, C. Schenken, J. Robinson, S. Williams, K. (2000) *ASP 3.0, Programmer's Reference*. Wrox Press Ltd.

Connolly, T. M. & Bygg, C. E. (1999) *Database Systems. A practical approach todesign, implementation, and management*. Andra upplagan, Addison Wesley University of Paisley.

Ek, J. & Arvidsson, S. (1999) *Active Server Pages och databaser på Internet.* Pagina Förlags AB. Göteborg: Elanders/Graphic Systems AB

Gundavaram, S. (1996) *CGI Programming on the world wide webb*. First edition. Sebastopol, Calif. : O'Reilly

Ju, P. (1997) *Databases on the web, Designing and Programming for Network Access*. MIS:Press & M&T Books.

Ladd, E. & O'Donnell, J. (1998) *HTML, JAVA och CGI*. 1998 Prentice Hall Europe. Pearson Educaton Limited

Lindman, B. (1995) *Informationsteknologi: Grundboken om modern datateknik.* Stockholm : Helsingfors : Pagina

Patel, R. & Davidson, B. (1994) *Forskningsmetodikens grunder : att planera, genomföra och rapportera en undersökning.* 2. uppl. Lund : Studentlitteratur

Ronne, E. (2000) *Avancerad Pocket, ASP, Active Server Pages.* Andra upplaga. Docendo Sverige AB 2000, Stockholm.

Silberschatz, A. & Korth, H. F. (1991) *Database system concepts.* Second ed. New York : McGraw-Hill

Bilaga A

Länkar till kod-exempel för CGI

http://hotwired.lycos.com/webmonkey/99/26/index4a_page2.html?tw=programming

http://hoohoo.ncsa.uiuc.edu/cgi/overview.html

http://www.15Seconds.com

Länkar till kod-exempel för ASP

http://www.15Seconds.com/

http://hotwired.lycos.com/webmonkey/98/39/index2a.html