COMPUTERWORLD

Sikker kodning af Web systemer og applikationer

Det er let at begå grundliggende fejl når man koder dynamiske web sites og applikationer. Artiklen giver en række eksempler på ting du bør tænke på og ting du bør undgå. Det meste er totalt ugenialt og helt igennem sund fornuft

Skrevet den **03. Feb 2009** af **bufferzone** I kategorien **Programmering / (D)HTML** | ★★★☆☆

Svage passwords, test konti og default konti

Konti oplysninger i form af brugernavne og passwords er væsentlige elementer i sikkerheden på de fleste elektroniske systemer, dette både når brugernavne og passwords giver logon direkte i applikationen, f.eks. i et forum, og nå passwords og brugernavne anvendes i logon på web hotellet eller til administration af systemet.

Svage passwords, f.eks. korte passwords, passwords der er lette at gætte eller passwords der står i ordbøger er meget lette at knække. Der findes Brute force programmer der kan skyde op til 300 forskellige passwords af i sekundet mod et system. Er passwordet kort, f.eks. under 8 tegn eller står passwordet i en ordbog, vil det hurtigt være brudt.

Default konti til den indledende konfiguration har jo af naturlige årsager kendte værdier. Enhver kan downloade dokumentationen og læse disse og derefter logge på med administratorrettigheder, hvis ikke disse konti er fjernet, disablet eller ændret.

Ofte anvendes test konti under opbygningen af disse sites. For at lette testn gives disse konti altid korte og naturlige passwords, som f.eks. ordet **test**. Ydermere ser man ofte at disse oplysninger direkte postes i forskellige fora for at få testes applikationen mere bredt.
Testkonti bør altid fjernes når testen er afsluttet, og man bør være tilbageholden med direkte at poste disse konti oplysninger i offentlige fora.

Færdige web applikationer

Mange bruger færdige løsninger der kan downloades gratis fra nettet. Disse applikationer er ofte både veludbyggede, funktionelle og sikre hvis man bruger dem fornuftigt.
Hovedproblemet ligger i defaultindstillinger, sample filer og test scripts. Default indstillinger dækker over navngivning og placering af kritiske filer som f.eks. log- og password filer, der, hvis de ikke fjernes eller ændres potentielt kan afslører kritiske ting.
Sampel filer og testscripts er en del af mange avancerede web løsninger. Disse giver ofte funktionalitet der kan anvendes ondsindet, og da alle har adgang til applikationerne kan både placering, funktionalitet og indstillinger forberedes før angrebet sættes ind.¨
Det er meget vigtigt at default indstillinger ændres, at sample og test scripts og test filer fjernes før systemet åbnes ud mod nettet.

Administrativ funktionalitet

At administrativ funktionalitet er kritisk er helt åbenbart. Det burde derfor også være helt åbenbart at administrativ funktionalitet bør beskyttes særligt omhyggeligt. Filer bør placeres i i biblioteker der er beskyttet med passwords og brugernavne og navngivningen af fil strukturen skal ikke være lette at gætte, da de så hurtigt vil blive fundet og udsat for angreb. En Placering som f.eks. /admin er helt almindelige og ualmindelig dum.

Forkert brug af robots.txt

Robots.txt er beregnet til at fortælle søgemaskinerne hvad de ikke skal indexiserer, ikke til at sørge for at

fortrolige informationer ikke bliver afsløret, Fortrolige informationer bør aldrig befinde sig på en web server, og de kritiske informationer man er nødt til at have, skal beskyttes aktivt, ikke skjules, får de bliver altid fundet.
En robots.txt fil kunne f.eks. se således ud:

```
User-agent: *
Disallow: /sysadmin/
Disallow: /sysadmin.asp
```

Vi ser her at biblioteket sysadmin og filer sysadmin.asp listes og dermed fortæller søgemaskinerne at de skal forbigå disse. Pointen er at søgemaskinerne ikke selv forsøger at gætte sig frem til filer og biblioteker, de søger kun i filer og biblioteker de kan finde via links og via linkstrukturen., hvis du ikke nævner biblioteket sysadmin i dine koder, herunder særligt i links så finder søgemaskinen ikke biblioteket og har du oven i købet passwordbeskyttet biblioteket, så er der slet ingen grund til at nævne dette i din robots.txt.
Hackere kan også læse og vil ofte i deres indledende undersøgelser kikke i robots.txt for at finde de steder du forsøger at skjule

inputvalidering i scripts
Input kan grundlæggende valideres på to forskellige måder. Man kan positivt validere eller negativt validere og man bør anvende begge former.
Positiv validering vil sige at man undersøger om inputtet indeholder de ting man forventer, det kunne f.eks. være et @ i en e-mail adresse. Hvis ikke @'et er til stede får man ikke lov til at submitte. Der er masser af muligheder for at validere positivt.
Negativ validering vil sige at man undersøger om inputtet indeholder ting det ikke må, det kunne f.eks. være mellemrum, tal, kodetegn, specialtegn m.m. Især kodetegn bør forbydes. Du kan også validere på antallet af tegn. Det er f.eks. sjældent du finder fornavne med mere end 20 bogstaver. Tastes mere ind, bør inputtet droppes.
Negativ validering kan også foretages på filtyper. Hvis du f.eks. driver et billedegalleri med mulighed for upload af billeder fra brugernes hjemmecomputere via nettet, er det en god ide at validerer på filtyper så der kun kan uploades f.eks. .gif og .jpg filer. Gør du ikke dette kan man f.eks. uploade en .asp fil der er kodet således at den sletter hele dit site, eller defacer alle dine sider. Mulighederne er uendelige. Kan der uploades og køres .exe filer så kan man lige så godt selv slette filerne og poste adgangskoderne på alt.hacker

Hvis du undlader at inputvaliderer eller inputvaliderer løst, vil bl.a. SQL injection være muligt. Her er et eksempel på dette:

Et script genererer f.eks. URL'en http://www.mitsite.dk/artikel.asp?ID=12 der viser artikel nummer 12 som hentes fra en database.
Værdien af feltet ID valideres i dette tilfælde ikke, men bruges ukritisk i en database forespørgsel og kan udformes manuelt som man lyster, det er kun fantasien der sætter grænsen.

Den bagvedliggende database forespørgsel ser ud som følger:

SELECT * FROM Artikel WHERE ID=12

Problemet er at værdien af ID kan ændres til noget andet således at kaldet f.eks kunne se ud som følger:

artikel.asp?ID=12;DROP TABLE Artikel

Der vil generere dette kald til databasen:

SELECT * FROM Artikel WHERE ID=12;DROP TABLE Artikel

Ovenstående kald vil kunne **SLETTE** artikel 12 fra databasen, såfremt rettighederne giver mulighed for dette, og hermed give hvem som helst mulighed for manuelt at tømme databasen for artikler, hvis de skulle få den lyst. Sletning er blot en at de mange muligheder SQL statements giver.

Validering af indirekte input
Input kan komme andre steder fra end direkte fra brugerne i form af indtastningsfelter eller URL'er

Når man udvikler skal man være opmærksom på at input ikke nødvendigvis kun er det der kommer direkte fra brugeren i form af indtastningsfelt eller en URL. Et godt eksempel kunne være cookies. Mange regner med at når informationen kommer fra en cookie, så er de sikre og kan bruges direkte i for eksempel et database kald. Denne antagelse er både forkert og farlig. Det kræver nemlig ikke den store viden at forfalske en Cookie og lave SQL injection gennem denne. Det kunne foregå på denne måde:

Hvis f.eks. en applikation styre/begrænser forskellige brugerrettigheder via en variabel der sættes til forskellige værdier således :
Cookie: brugerrettigheder=10

Kan man modificerer/neutraliserer de satte begrænsninger med denne modificerede cookie:
Cookie: brugerrettigheder=10 or 1=1

Hvis f.eks. en applikation anvender et log system der i en database logger hvilke typer browsere brugerne på et website benytter. Den normale kode kunne f.eks. se således ud:

User-agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)

Hvis en person med onde hensigter modificerer koden således:;
User-agent: ');exec master..xp_cmdshell 'del *.* /q';
Vil han via exec master..xp_cmdshell, der er en standard stored procedure kunne slette hele databasen eller det der er værre.

Benytter man HTTP headers eller cookies i sine scripts bør man også validere dette input.

Include filer uden server side association
Include filer bruges ofte når man laver dynamiske sites, det er en bekvem måde at genbruge kode på og derved spare både kodnings tid og fejlmuligheder. Et ekspempel på en include file der ofte bruges er en fil der indeholder den kode der laver forbindelsen ned til databasen i dynamiske sites. Inkluderingen af filen kunne se således ud:

```
.....
<!--#include virtual="/includes/opendatabase.inc"-->
.....
```

Ved første øjekast ganske smart og problemløst, men ved nærmere eftertanke er der faktisk et problem. Den fil vi inkluderer har endelsen .inc. Denne endelse er ikke associeret til noget som helst, og hvis en hacker linker direkte til filen vil indholdet, og dermed vores forbindelseskode vises i browseren.
Hvis vi ønsker at undgå dette, kan vi bruge en endelse der er associeret med en fortolker på web serveren, f.eks. .asp.

Man bør benytte fil endelser der er associeres med en fortolker alle de steder det kan lade sig gøre, altid

på filer, der indeholder kode, således at kildekode, database struktur eller anden fortrolig information ikke afsløres i det tilfælde at f.eks. include filen bliver kaldt direkte.

Hvis f.eks. en hacker i ovenstående tilfælde skriver http://www.mitsite.dk/includes/opendatabase.inc, så vil browseres servere koden herunder.

```
<%
'Denne fil åbner forbindelsen til databasen
strconn = "Driver={SQL Server};SERVER=sqlserver;UID=sqluser;PWD=sapasswd;DATABASE=kunder
set conn = server.createobject("adodb.connection")
conn.open strconn
%>
```

Backup filer
At tage en backup af en script fil før man retter i den, er altid en god ide. At beholde denne fil på systemet, med et let gætteligt navn og placeret uden beskyttelse, er altid en dårlig ide, da disse filer ofte kan afsløre kildekode. Backup filer bør placeres uden for web serverens rækkevidde eller endnu bedre på flytbare medier.
Eksempler på forkert brug af backup filer:
default.asp.old
default.asp.slet
default.asp2
default.bak
etc.
Eksempler på korrekt brug af backup filer:
backup_default.asp
sendmail.old.pl
Disse filer er begge associeret med fortolkere (perl og ASP fortolkeren), de vil blive fortolket og ikke vist hvis de kaldes direkte

Nogle HTML editeringsprogrammer efterlader automatisk en backup fil der hvor de arbejder. Hvis man editerer direkte på sin webserver, skal man efterfølgende være opmærksom på at der kan ligge kopier af alle de filer man har editeret.

Stone's Webwriter f.eks. efterlader .bak filer og Linux editoren VI tilføjer en tilde (~) til det oprindelige filnavn. Begge disse filer vil afsløre kildekoden, hvis de kaldes direkte i en browser, især Stone's kan give problemer idet den ændre ekstensionen til bak, hvilket kan afsløre indholdet af filer der ellers ville blive fortolket.

Eksempler på backupfiler:
Filen default.asp editeres med Stone's Webwriter eller VI.
Efterfølgende befinder der sig nu en backup fil i webtræet.

Efter Stone's Webwriter
default.asp
default.asp.bak

Efter VI
index.php
index.php~

unødvendige information i HTML eller JavaScript kode.
Hvis man anvender færdige systemer, eller JavaScript der er downloadet fra diverse gratis script biblioteker på nettet, vil man ofte kunne finde detaljerede oplysninger som kommentarer i koden. Kommentarer skal altid gennemses for oplysninger der kan anvendes eller hjælpe en hacker. Det kunne f.eks. være versioneringsoplysninger, der kunne give en hacker mulighed for at finde sårbarheder og exploits på nettet, der kan anvendes til at komme ind. Det kunne være stioplysninger, der kunne give en hacker mulighed for at tilgå filer direkte eller det kune være navne oplysninger, der kunne hjælpe en hacker med at gætte brugernavne og evt. passwords. Fjern for en sikkerhedsskyld så meget at kommenteringen som muligt fra systemer der er i drift på nettet

Forms og hidden fields
Hidden fields benyttes til mange forskellige ting i forms. Det at de er hidden betyder ikke automatisk at felterne ikke kan manipuleres.

Hvis du f.eks. bruger et hidden field til at indeholde den e-mailadresse informationerne fra formen skal sendes til og indholdet af dette felt kan manipoleres, vil din applikation måske kunne bruges til at udsende spam med.

Hvis du f.eks. har en e-shop og bruger hidden field til at indeholde oplysninger om valuta, kan du sikkert regne ud at det har en vis betydning om valutaen er i danske kroner eller i Euro. Den ene vej vil give meget salt og negativ indtjening, den anden vej vil give ingen salg og ingen indtjening. Begge dele potentielt ødelæggende for din forretning .

Eksempel på hidden fields:

```
<form action="sendmail.cgi" method="post">
<textarea name="tekst"></textarea>
<input type="hidden" name="recipient" value="kim@bufferzone.dk">
<input type="Submit" value="Kontakt mig">
</form>
```

Var det så det
Der er helt sikkert flere fejl og dumme ting du kan gøre når du koder dit dynamiske web site. Har du forslag til ting der skal med, kommentarer eller rettelser (f.eks. stavefejl) er du meget velkommen til at kontakte mig på kim@bufferzone.dk ligesom jeg ofte er at finde her på eksperten.
Jeg vil bede dig om at undlade at stille spørgsmål i artiklens kommentarfelt, da jeg jo ikke her kan besvare dem.

Kommentar af a1 d. 06. Feb 2006 | 1

fint artikel, god for begyndere...
man skal dog altid validere et "id", ala id = Cint(Request.Querystring("id"))

Kommentar af lsskaarup d. 06. Feb 2006 | 2

Se det er en artikel man kan bruge til noget.

Kommentar af coder d. 05. Jan 2008 | 3

Den burde vel ligge under "Sikkerhed" og ikke (D)HTML

Kommentar af krab d. 10. Nov 2006 | 4

Spændende og brugbar

Kommentar af the_email d. 05. Feb 2006 | 5

God artikel, med mange overvejelser som jeg ikke selv havde gjort mig. Thumbs up!

Kommentar af cyberjelle d. 08. Feb 2006 | 6

Udemærket (ikke mere)
Selvom dine artikler plejer at være virkelig nyttige, er denne helt klart ikke en af dine bedste. Synes den vinder for god karakter - udelukkende på bufferzones image. Alligevel dog 5 points værd ;)

Kommentar af roenving d. 05. Feb 2006 | 7

Kommentar af konder d. 09. Feb 2006 | 8

Kommentar af osborne d. 02. Jan 2008 | 9

Rigtig god artikel

Kommentar af jih d. 06. Feb 2006 | 10

mange gode råd, som helt sikkert kan bruges til noget, såfremt man - som mig - ikke ved meget om dem.. :-)

Kommentar af dustie d. 24. Nov 2006 | 11

Kommentar af saudoo d. 07. Jan 2008 | 12

Kommentar af over-load d. 04. Aug 2006 | 13

!

Kommentar af wicez (nedlagt brugerprofil) d. 05. Feb 2006 | 14

Forholdsvist basale dog meget nyttige ting du kommer ind på. Fremragende artikel bufferzone - endnu engang.

Kommentar af swiatecki d. 05. Feb 2006 | 15

Kommentar af madeindk d. 01. Feb 2007 | 16

Glimrende, godt skrevet bufferzone.

Kommentar af venchil d. 05. Feb 2006 | 17

Jeg er selv ved at udvikle i øjeblikket, og opdagede, selvom jeg kun er 100 linjer inde i koden fejl og mangler. Tak.

PS: Der er et par stavefejl

Kommentar af benefaktor (nedlagt brugerprofil) d. 14. Feb 2006 | 18

Kommentar af mr-kill d. 06. Feb 2006 | 19

Kommentar af cirrhosis (nedlagt brugerprofil) d. 12. Feb 2006 | 20

God introduktion til sikker udvikling. Burde være gratis og have sit eget sticky link/banner, så flere ville læse den, da den kommer i dybden med typiske fejl man finder overalt på nettet. Gode eksempler.

Kommentar af lassemelbye d. 05. Jan 2007 | 21

Fantastisk -- helt sikkkert 5 points værd. Enhver webudvikler bør læse dette

Kommentar af 4262sandved d. 28. Jun 2006 | 22

Rigtig god artikel.

Kommentar af happycow d. 04. Jan 2008 | 23

Dækker en del -- For en newb er den 5 point værd

Kommentar af gvp d. 04. Mar 2008 | 24