HTML5 UR ETT SÄKERHETSPERSPEKTIV

ETT FÖRSTA UTKAST – KOMPLETT MED SAKNADE KÄLLHÄNVISNINGAR OCH STAVFEL :)

Magnus Gulin

Kandidatavhandling i datavetenskap

Institutionen för informationsteknologi vid Åbo Akademi

Handledare: Frank Wickström

Våren 2013

Referat

Utvecklingen av Internet, och då särskilt World Wide Web, går framåt i en rasande takt. När även banktjänster och andra personliga uppgifter hanteras på nätet, är det viktigt att de hanteras säkert. När webbläsare uppdateras för att hantera nya protokoll och språk medför det också att nya säkerhetshål introduceras. HTML5 medför både nya funktioner som kan utnyttjas för illasinnade aktiviteter, men också funktioner som skall skydda användaren. Med min kandidatavhandling vill jag undersöka dessa för- och nackdelar med HTML5.

INNEHÅLLSFÖRTECKNING

INLEDNING

I HTML5 finns många nya funktioner och möjligheter att bygga hemsidor på ett sätt som gör att de mera påminner om desktopprogram. Denna utökade funktionalitet gör också att webbläsaren har svårare att veta A) om information/kod den har mottagit från en källa är pålitlig, och b) kan den skicka denna information vidare till en annan källa. HTML5 innehåller säkerhetsfunktioner som gör att webbläsaren ”vet” hur den skall hantera dylik information, och därmed stoppar attacker som webbkodsinjektion (*Cross-site Scripting*, XSS), *Cross-site Request Forgery* (CSRF) och klicknappning (*Clickjacking)*.

På Åbo akademi har vi flertalet kurser där vi skapar webbapplikationer, men ingen av dessa lär ut hur man sedan ”stryper ner” sin applikation för att undvika att användarna drabbas av t.ex stulna lösenord. Detta är enligt min mening en bra motivering för att skriva om just detta.

TODO: Hitta svenska uttryck, skriv längre, skriv seriösare

VAD ÄR HTML

Hypertext Markup Language, hädanefter HTML, är ett så kallat märkspråk, och används för att bygga upp layout och funktionalitet på webbsidor. Första versionen av detta kom ut 1995. Språket har sedan uppdaterats i flera omgångar, och har därmed fått mera funktionalitet.

Nu för tiden används ofta HTML för att strukturera material, medan själva layouten byggs upp med hjälp av stilmallar. Standarden dessa använder kallas Cascading Style Sheets, hädanefter CSS.

Förutom att strukturera och visa material kan man på webbsidor också köra små program, så kallad *webbkod*. Webbkod introducerades först under namnet Mocha, och döptes senare till Javascript. Microsoft gjorde sedan sin egen implementation som döptes till Jscript. Språket blev standardiserat i juni 1997 under namnet ECMAscript. Alla webbläsare strävar till att vara kompatibla med ECMAscript, men kan också innehålla utökad funktionalitet. ECMAscript har också detta uppdaterats i omgångar, och på grund av dess rika funktionalitet används det i majoriteten av säkerhetshålen jag kommer att behandla. För att undvika förvirring mellan de olika implementationerna används hädanefter begreppet webbkod.

De flesta webbläsare har dessutom stöd för plug-in program. Dessa utökar webbläsarens funktionalitet och kan t.ex. visa proprietära filformat. Vanliga plugin är t.ex. Java applets, Silverlight och Flash.

TODO: Leta upp källor för alla påståenden

NYHETER I HTML5

HTML5 introducerar flera nya taggar för multimedia, så som <audio>, <video> och <canvas>[1]. Med hjälp av dessa behöver webbläsaren inte längre plug-in program för att kunna hantera multimedia, och konkurrerar därmed med dessa program.

POSTMESSAGE [2]

Postmessage är en funktion för att kommunicera mellan olika domäner. På en webbsida kan man inkludera en eller flera ramar, och sedan sända och ta emot data mellan sidorna i dessa och själva sidan med ramarna. PostMessage erbjuder också filtrering så att den mottagande sidan kan bestämma att data endast får tas emot från en viss värd.

OFFLINE WEB APPLICATIONS

TODO!

LOCAL STORAGE

Local storage är egentligen inte bundet till HTML5-specifikationerna. Istället har det ett eget specifikationsdokument på TODO.

Local Storage är ett mera versatilt alternativ till cookies.

WEB SOCKETS [3]

Med detta får webbläsaren (och därmed webbsidor) möjlighet att använda TCP-sockets för kommunikation. Därmed kan webbsidor och http-servrar kommunicera på ett mera fritt sätt än tidigare. Web sockets är inte heller bundet till just HTTP-klienter och -servrar, utan kan användas av vilka klienter och servrar som helst.

HOT I HTML

WEBBKODSINJEKTION (CROSS-SITE SCRIPTING, XSS)

Webbkodsinjektion innebär att en attackerare utnyttjar säkerhetshål i en webbapplikation för att injektera skadliga HTML-taggar som sedan visas åt andra besökare. Eftersom denna kod visas på den attackerade webbsidan kan den ändra på hemsidans struktur och till exempel skicka användarnas formulärdata och autentiseringskakor till attackeraren.
Koden kan endera omges med <script>-taggar, eller så kan man utnyttja attribut till andra htmltaggar. Följande kod visar exempel på olika attackvektorer. Koden alert(document.cookie) brukar användas för att demonstrera webbkodsinjektion, eftersom den visar sidans kakor i ett dialogfönster.

```
<script>alert(document.cookie)</script>
<body onload=alert(document.cookie)>
<b onmouseover=alert(document.cookie)>click me!</b>
<img src=http://example.url onerror=alert(document.cookie);>
```

Första raden visar hur webbkodsinjektion fungerar på enklaste tillvägagångssättet. Detta fungerar dock inte om < och > filtreras bort. Rad 2, rad 3 och rad 4 kräver inte dessa tecken.

Webbkodsinjektion kan delas upp i tre kategorier: [2]
1. Icke-bestående webbkodsinjektion
Den injekterade koden sparas aldrig på servern, utan kommer till exempel via länken. Koden når alltså endast de besökare som anländer via den preparerade länken. Ett exempel på detta är sökfunktioner som också visar vilken term man sökt efter.

2. Bestående webbkodsinjektion
Koden sparas på servern, och man behöver ej klicka på en preparerad länk för att se detta. Ofta visas koden också för alla webbsidans besökare. Ett exempel på detta är kommentarsfunktioner på webbapplikationer så som bloggar.

3. DOM-baserad webbkodsinjektion
I denna kategori kommer den injekterade koden aldrig att skickas över nätverket, istället så används redan existerande webbkod för att skriva ut den skadliga koden från exempelvis urlen.
Exempel: I webbsidan finns koden

```
<SCRIPT>
var pos=document.URL.indexOf("name=")+5;
document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
```

Normalt sett skulle denna kod användas för att välkomna användaren:
http://www.vulnerable.site/welcome.html?name=Joe
Men skulle också kunna användas för kodinjektion:
http://www.vulnerable.site/welcome.html?name=<script>alert(document.cookie)</script>

Faktum är att koden som injekteras behöver aldrig över huvud taget skickas till servern. För mera info, se [3].

CROSS-SITE REQUEST FORGERY CSRF

Precis som namnet säger så skickas en förfrågan från en sida till en annan. Som exempel kan vi ta en legitim webbapplikation, där användaren är inloggad. Applikationen har ett formulär för att skapa nya användare, och skickar data från formuläret till servern. En attackerare kan skapa en färdigt ifylld kopia av formuläret på sin webbsida, som dessutom kommer att skickas automatiskt. Om användaren luras att besöka sidan med det falska formuläret kommer dennes webbläsare att skicka in formuläret till webbapplikationen, och därmed skapa en användare. [3]

KLICKNAPPNING (CLICKJACKING)

Detta är ett förhållandevis nytt hot, som bygger på CSS. CSS ger möjlighet att bygga HTML-sidor i 3 dimensioner (x-, y- och z-index). På skärmen syns sedan endast två dimensioner. Attackeraren gör en webbsida som har något oskyldigt i det översta lagret, och är således det som syns åt användaren. När användaren klickar på detta så överförs klicket från det översta lagret till ett osynligt lager undertill. Detta kan exempelvis vara en ram med en annan webbsida.[4]

HOT SOM HAR KOMMIT TILL I HTML5

Eftersom HTML5 bjuder på många nya funktioner ökar också risken för att dessa skall missbrukas. Exempelvis måste filter för webbkodsinjektion uppdateras för att filtrera bort nya html-taggar och -attribut. HTML5 har också möjlighet att påverka webbläsarens sidhistoria, d.v.s. vilken sida man kommer till om man använder tillbakaknappen.

Klicknappning kan vara lättare att utföra med HTML5. Normalt sett skyddar man sig mot detta genom att ha webbkod på sin webbsida som gör att sidan bryter sig ut ur ramar. I HTML5 får <iframe>-taggen ett nytt attribut, *sandbox*. Detta gör att man kan skapa en ram som inte har rätt att köra kod. I detta fall är det dock negativt, eftersom en attackerare kan använda detta för att motarbeta utbrytningskoden.

Klicknappning har fungerat för att lura användare att klicka på saker, men om man till exempel vill lura en användare att beställa hem en vara åt sig måste denna fylla i en adress manuellt. HTML5 har stöd för drag-and-drop-funktionalitet, d.v.s. att en användare kan dra ett element från en plats till en annan. Detta kan vara till exempel text som man drar till en textruta. Drag-and-drop-funktionaliteten i samband med klicknappning kan också utnyttjas för att få användaren att ofrivilligt dela med sig av privat information från en webbsida till en annan.

HTML5 har nya funktioner för att spara data lokalt (Local Storage). Om en webbsida har kodinjektionshål så kan en attackerare eventuellt använda sql injection mot detta. Det finns också en risk för att en attackerare kan använda upp allt hårddiskutrymme, om inte webbläsaren har någon gräns för detta.

Normalt sett har det varit omöjligt för webbkoden på en server att kommunicera med en helt annan server. Detta på grund av Same Origin Policy. HTML5 introducerar en funktion vid namn CORS, vilket tillåter att sida A gör anrop till sida B. Detta skyddas visserligen av att sida B måste tillåta detta i sin HTTP-header, men detta skyddar endast mot att sida A inte kan läsa data från sida B. Man kan således fortfarande göra request som exempelvis gör inköp eller skapar en ny användare. ←TODO KÄLLA? Enligt NG_web_security så görs först en pre-flight request som kontrollerar om servern tillåter denna host att skicka förfrågningar. TODO Tidigare källa har yrat ihop simple requests(utan preflight) och actual requests(kräver preflight).

Web sockets är en stor nyhet i HTML5. Detta möjliggör att man kan programmera vilka internetprotokoll som helst i sina webbsidor. Detta kan förstås också användas för illvilla ändamål, så som att undersöka användarens lokala nätverk och utnyttja säkerhetshål där.

HTML5 innehåller också Web notifications, som tillåter att webbsidor skapar ”popups” utanför webbläsaren, precis som vanliga program. Dessa kan också utformas med html-kod.

Detta är ingen direkt säkerhetsrisk, men kan underlätta för en attackerare att härma systemprogram och lura användare till att t.ex. klicka på länkar som leder till illvilliga webbsidor.

HUR MAN KAN SKYDDA SIG MOT HOT I HTML

HJÄLPMEDEL SOM KOMMIT MED HTML5

FRAME SANDBOXING

I HTML5 har ramar (dvs <frame>-taggen) fått ett nytt attribut, *sandbox*. Om man i sin webbapplikation måste inkludera icke trovärdiga webbsidor i ramar kan man med detta attribut förbjuda att webbsidan i ramen exekverar webbkod.
Detta kan tyvärr också användas för illvilliga syften, eftersom webbsidor ofta använder webbkod för att förbjuda att de visas i ramar.

CROSS-ORIGIN RESOURCE SHARING (CORS)

Flera nya HTTP headers har införts i HTML5. Med hjälp av dessa kan man specificera vilka domäner som skall kunna göra förfrågningar till sin egen sida. TODO detta skriver jag om i ”nya hot”. Planera om ordningen av dessa.

Access-Control-Allow-Origin

POSTMESSAGE

todo

SAMMANFATTNING

I och med HTML5 förändras sättet vi kan använda webbsidor och webbapplikationer. Från att man tidigare har varit tvungen att använda plugin-program så som Flash klarar man sig nu på endast HTML-kod. Dessutom finns det nu mera möjligheter att dela med sig av data mellan olika webbsidor, vilket är mycket bra.

Tyvärr leder utökad funktionalitet också till fler säkerhetshål. Det känns som att tyngdpunkten ligger på att skapa mera funktionalitet istället för att se till att det man har fungerar säkert. Läget är trots allt inte så illa, eftersom HTML5 knappast är en revolution, utan mera av en evolution.

Definitioner: TODO, stoppa in dessa i texten före de används?

Användare: Användare är en person eller flera personer som använder en webbläsare för att besöka (legitima) webbsidor och använda webbapplikationer.

Attackerare: Attackerare är en eller flera personer som försöker utnyttja säkerhetshål för att få användarnas personliga information eller motsvarande.

Webbsida: En sida skriven i HTML, eventuellt med utformning gjord i CSS. Kan också innehålla webbkod och övrig media.

Webbapplikation: Med webbapplikation menas en portal gjorda av flera webbsidor. Webbapplikationer har också funktionalitet på serversidan, exempelvis databaser med användarnas information.

Domän: I betydelsen domännamn. Används istället för engelskans domain.

BIBLIOGRAPHY

[1] R. Clark, O. Studholme, C. Murphy and D. Manian, Beginning HTML5 and CSS3, Apress, 2012.

[2] "HTML5 specifications," [Online]. Available: http://www.w3.org/TR/2012/CR-html5-20121217/. [Accessed 26 02 2013].

Todo:

Hitta svenska ord för: Cross-site request forgery, url, HTTP-header,

Byt ut iframe och frame mot ramar, där det går.

Skriv om säkerhetshål med local storage, samt xss-hål som lämnar kvar med local storage och persistent applications.

Skriv eventuellt om sql injection på servrar?

Skriv mera exempelkod för varje attackvektor.

Color highlightning på kodsnuttarna