[OFULLSTÄNDIG]

Testning av webbapplikationer för Cross-site scripting

Wiljam Tschernij, 39098
Kandidatavhandling i Datateknik
Handledare: Dragos Truscan

Institutionen för informationsteknologi
Åbo Akademi
2019

Referat

En webbsida är ofta en organisations front mot omvärlden – och internet – vilket gör den känslig mot utomstående attacker av många slag. Webbapplikationer är applikationer som kör i användarens webbläsare, och kommunicerar med en server. När en webbapplikations inmatning inte saneras (dvs. analyseras och rengörs av suspekta karaktärer) kan en angripare injicera programkod som körs på webbsidan, och eventuellt också datorn av offret. En specifik instans av dessa injektionsattacker är Cross-site scripting-attacker (XSS), vilka denna avhandling hanterar.

Ett sätt att försvara sig mot denna sortens attacker är att testa webbapplikationen före den ställs ut på internet. Mjukvarutestning går ut på att testa ett programs funktioner, både statiskt (en funktion i taget), samt dynamiskt (vid programmets körtid). En överskådlig blick om mjukvarutestning kommer att ges i avhandlingen, samt hur mjukvarutestning kan tillämpas för att skydda webb-applikationen mot XSS-attacker.

Innehållsförteckning

1. Introduktion

I begynnelsen var internet endast ett nätverk där man rent tekniskt endast kunde ladda ner filer med text i. Då man öppnar sin webbläsare idag är det mycket svårt att hamna på en webbsida som inte har någon annan funktion än att visa användaren text. Även de enklaste webbsidorna har idag åtminstone en inloggningsfunktion, om inte ett kommentarfält eller en feedback-funktion. I början av 2000-talet när webbsidorna började växa och fick mera funktionalitet, började de upprätthålla en uppkoppling till ursprungsservern, den sk. webb-servern. Det viss säga, istället för att ladda ner en HTML-fil från servern och avsluta, så fortsatte webbsidan kommunicera med webbservern med hjälp av nerladdad programkod.

Om man betraktar ordet "webbsida", ser man dess ursprungliga mening: en sida av text som man hämtar från "webben", dvs. internet. Under tidens gång har ordet "webbsida" fått en ny mening, alltefter "webbsidor" – alltså då webbapplikationer – har fått mera funktionalitet.

1.1. Sårbarheter i webbapplikationer

1.2. Om webbapplikationer

Någonting vi idag kallar en webbsida är egentligen en webbapplikation. En webbapplikation är olik en webbsida i hur de fungerar inom webbläsaren: då en webbläsare hämtar en webbsida, i traditionell mening, från en server, hämtar den filer i två format; oftast en fil i HTML-format (Hyper Text Markup Language) och filer i CSS-format (Cascading Style Sheet). HTML-filen berättar för webbläsaren vad som finns på sidan (stycken, rubriker, listor med text), och CSS-filen hur den ska lägga upp webbsidan så att den ser ut så som webbsidans skapare bestämt (färger, typsnitt, osv.).

En webbapplikation har en mer aktiv roll: efter att ha laddat ner sitt innehåll till klienten kommer webbsidan att hålla kontakt med den associerade servern. Webbsidan innehåller alltså kod som reagerar på användarens handlingar, och därmed skickar samt mottar data från servern. Vid mottagen information svarar servern åt klienten eller sparar det mottagna datat i en databas. Om datat i fråga inte analyseras och saneras (eng. sanitized) före det sparas eller skickas tillbaka av servern kan en angripare injicera kod i datat, vilket oftast inte är idealiskt.

Injiceringen av kod introducerar säkerhetsrisker för både användare av webbsidor och organisationen som upprätthåller webbservern. Med

Det finns även webbapplikationer som inte kommunicerar med servern, utan laddar ner allt den behöver på första gången. Dessa brukar kallas "native" applikationer, eftersom de kör på användarens dator. Den sortens applikationer är inte i allmänhet sårbara för XSS-attacker eftersom det inte skickas någon data mellan servern och klienten efter den första gången.

Webbapplikationer har inte formaliserats akademiskt men dess arkitektur omfattar alltid en klient och en server. Det finns ingen officiell standard för webbapplikationsarkitekturer och detta har troligen sina rötter i det att webbapplikationer ofta är tätt kopplade med affärsverksamhet. I affärsvärlden – under tidiga 2000-talet då webbapplikationer först uppkom – är det ofta en fördel att inte avslöja sina hemligheter åt de tävlande parterna.

1.3. Godtycklig inmatning

För att vara användbar måste en webbapplikation tillåta en användare att mata in text. Ett program som inte låter användaren mata in någonting, är inte ändamålsenligt under de flesta omständigheter. Även en webbapplikation som enbart tillåter användaren att kontrollera information, om den på något sätt är känslig, kräver ett inmatningsfält för användarnamn och lösenord.

Det unika problemet med webbapplikationer är att de är tillgängliga för vem som helst med tvivelaktiga motiv – eftersom de flesta är tillgängliga på internet – och därav är mer sårbara än program som enbart körs på användarens lokala dator. Som ett exempel måste webbsidan för en bank ge användaren möjligheten att mata in åtminstone sitt användarnamn och lösenord för att ta hand om sina bankärenden, annars är själva webbapplikationen onödig.

2. XSS-attacker

2.1. Injiceringsattacker

När en sträng av något slag matas in i en dator, måste den tolkas på något sätt. Det är egentligen en av grundprinciperna när det kommer till beräkningar och logik – en dator kan begreppsligas som en låda men data som går in i den ena ändan och kommer ut i den andra. Denna tolkning kan ställa till med problem: den inmatade strängen kan tolkas fel av datorn kan verka på ett oavsiktligt sätt. När det kommer till datasäkerhet bör oavsiktliga konsekvenser alltid undvikas.

En inmatning hanteras ofta som en sträng av bokstäver av programkoden, där strängen avgränsas, i början och i slutet, av ett särskilt tecken, oftast av citattecken ("-"). Om man då tillägger två citattecken i inmatningsfältet, ett i början och ett i slutet av någon programkod som en illvillig aktör vill köra, kan man injicera (eng. inject) kod i programmet. Ett brott uppstår alltså i den tolkade inmatningssträngen och godtycklig kod kan köras.

De programmeringsspråk som används på klientsidan av webbapplikationer är just tolkade, kontra kompilerade. De är baserade på ECMAScript standarden; det oftast förekommande språket är JavaScript, med det finns andra språk som är baserade på ECMAScript [1].

XSS-attacker tillåter angripare att stjäla uppgifter om användaren: inloggningsinformation eller filer från offrets dator. XSS tillåter också en

inkräktare att skanna inifrån det nätverk som den ansatta datorn är kopplad till, vilket kan ge anfallaren access till andra datorer på nätverken.

2.2. Om XSS-attacker

XSS-attacker är en bred klass av attacker. Cross-site-scripting är en attacktyp vars attackvektor är länkar. Kan användas till att knyta samman andra svagheter i offrets

2.3. Reflekterade

2.4. Beständiga

(eng. Stored eller Persistent).

2.5. DOM-baserade

En Document Object Model (DOM) är en representation av ett dokument i datorns arbetsminne.

3. Programvarutestning och XSS-attacker

3.1. Allmänt om programvarutestning

Generellt sett är programvarutestning ett disciplin där man försöker finna brister i programkod för att undvika oavsiktliga sidoeffekter då programmet kör som produktionskod, dvs. kod som slutanvändarna interagerar med. Man kan lätt se hur programvarutestning går ihop med datasäkerhet: om man vill skydda sig emot möjliga cyber-angripare, måste man försöka sätta sig i deras skor och försöka få applikationen att gå mot skogen, på ett sätt eller annat.

Det finns flera sorter av testning man kan göra på ett program: de två huvudkategorierna är white-box- och black-box-testning. Dessa två kategorier har och göra med hur mycket testets skapare kan se själva logiken som utgör programmet. För att kunna göra ett white-box-test, måste testaren ha tillgång till

programmets källkod för att kunna analysera det. Det finns en orsak varför det kan vara lättare att göra black-box tester: realistiskt sett är det samma "vy" av programmet som en möjlig attackerare kommer att ha av det körande applikationen.

Eftersom denna avhandling hanterar XSS-attacker, kommer vissa testnings-metoder att inte tas i beaktande.

3.1.1. Testorakel

Ett testorakel (eng. test oracle) är den parten som avgör om ett test passerat eller inte. Dessa orakel kan vara mänskliga eller automatiserade system, beroende på domänen av det slutliga programmet som testas; det behövs automatiserade tester för att täcka basfallen, men för att upptäcka nya säkerhetsluckor, eller helt och hållet nya kategorier av dem, behövs en mänsklig aktör (åtminstone än så länge). När det gäller till testning av applikationer för specifikt XSS-sårbarheter, är de flesta orakel mänskliga, på grund av naturen av problemet: det hittas nya XSS-sårbarheter jämt.

3.2. Statisk kontra dynamisk testning

Det finns flera olika sätt att testa ett program. Ett sätta är att analysera programmet före det körs; detta kallas statisk testning.

3.3. Blackbox-testning

Blackbox-testning är det enda sättet att testa programvara till vilken man inte har tillgång till källkoden. Detta är också en favoritmetod för olegitima hackers (så kallade blackhat-hackers) att undersöka, eller potentiellt söndra eller ta in sig i, en webbsidas kod eller data på serversidan.

4. Diskussion

XSS-attacker är en rätt bred kategorisering av attacker. Dels är också medverkandet av alla tre parter i attacken (angriparen, offret och webbsidan) en faktor som leder till flera möjliga kombinationer. Detta är också varför denna sorters attacker är svåra att testa.

5. Referenser

[1] B. T. Allen Wirfs-Brock, ”ECMAScript® 2017 Language Specification (ECMA-262, 8th edition, June 2017)”, *ECMA International*, juni-2017. [Online]. Tillgänglig vid: https://www.ecma-international.org/ecma-262/8.0/index.html. [Åtkomstdatum: 13-feb-2019]

[2] M. I. P. Salas och E. Martins, ”Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services and WS-Security”, *Electron. Notes Theor. Comput. Sci.*, vol. 302, s. 133–154, 2014 [Online]. Tillgänglig vid: http://dx.doi.org/10.1016/j.entcs.2014.01.024

6. Bibliografi

D. Stuttard, M. Pinto, "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws", 2nd edition. Tillgänglig vid: https://abo.finna.fi/Record/alma.1108633

Open Web Application Security Project (OWASP), "OWASP Testing Guide 4.0". Tillgänglig vid: https://www.owasp.org/images/1/19/OTGv4.pdf