

Reflekterad Cross-Site Scripting

Filip Stenegren

MITTUNIVERSITETET

Institutionen för Informations-system och teknologi

Examinator: Mikael Hasselmalm, mikael.hasselmalm@miun.se

Handledare: Nayeb Maleki, nayeb.maleki@miun.se

Författare: Filip Stenegren, fist2000@student.miun.se

Utbildningsprogram: DT167G, 7.5 HP

Huvudområde: Datateknik

Innehåll

Reflekterad Cross-Site Scripting	1
Filip Stenegren.....	1
Terminologi.....	Fel! Bokmärket är inte definierat.
Akronymer/Förkortningar	Fel! Bokmärket är inte definierat.
1 Introduktion.....	1
2 Teori	2
2.1 Cross-Site Scripting (XSS).....	2
2.2 Vad är Reflekterad XSS	2
2.3 Hur fungerar Reflekterad XSS	2
2.3.1 Inputvalidering	3
2.3.2 Outputfiltrering	3
3 Metod	4
3.1 Projektprocess.....	4
3.1.1 Bakgrundsstudie.....	4
3.1.2 Identifiera XSS-sårbarheter.....	4
3.1.3 Avgränsningar	4
3.1.4 Implementering av XSS-sårbara hemsidan.....	4
4 Konstruktion.....	5
5 Resultat.....	6
6 Slutsatser	8

1 Introduktion

Reflected Cross-Site Scripting (XSS) är en mjukvarusårbarhet. Denna sårbarhet gör det möjligt för angripare att injicera illvillig kod på en webbsida som sedan körs på offrets webbläsare. Reflekterad XSS är en form av XSS där attackeraren utnyttjar sårbarheter i en webbsidas sökformulär, URL-parametrar eller andra inmatningsfält, för att injicera skadlig kod.

När offret sedan besöker den manipulerande sidan kommer den illvilliga koden köras på offrets dator. OWASP (Open Web Application Security Project) rankade även XSS som topp 7 i deras topp 10 lista 2017 [1]

Reflekterad XSS är ett allvarligt hot mot säkerheten på webben eftersom det kan leda till stöld och spridning av känslig information, spridning av skadlig programvara och kan till och med avlägset styra offrets webbläsare.

För att skydda sig mot reflekterad XSS kan webbutvecklare utnyttja tekniker som inputvalidering, outputfiltrering, säker hantering av sessionsvariabler och använda säkra http-headers. Utbildning och medvetenhet om säkerhetsriskerna associerad med reflekterad XSS är därför viktig för att minska risken för en lyckad attack.

I denna rapport kommer det diskuteras vad reflekterad XSS är och hur det fungerar, de vanligaste sårbarheterna som attackerare utnyttjar för att utföra reflekterade XSS-attacker samt de bästa metoderna för att skydda och stoppa dessa attacker.

2 Teori

2.1 Cross-Site Scripting (XSS)

Cross-Site Scripting är en typ av injektion där skadliga script injiceras i vanligtvis ofarliga och betrodda webbplatser. XSS-attacker uppstår när en angripare använder hemsidan för att skicka skadlig kod. Koderna skickas vanligtvis via....

Ett enkelt sätt att testa om en hemsida är sårbar för XSS är att i ett sökfält eller annat område som kräver användarinput skriva:

```
<script>alert("XSS")</script>
```

Svarar sidan då med vad figur 1 visar är sidan sårbar för XSS attacker



figur 1, Sårbar hemsida

Lyckade XSS attacker kan även åstadkomma att förändra utseendet och beteendet på hemsidor. [2][3]

2.2 Vad är Reflekterad XSS

Reflekterad XSS, även kallad Non-Persistent XSS, är en typ av webbapplikationssårbarhet som uppstår när en attack kan injicera skadlig kod i en vanligtvis pålitlig webbapplikation som sedan körs på en användares webbläsare. Reflekterad XSS skiljer sig åt lagrad XSS, där den injicerade koden lagras på en server och distribueras sedan vidare till andra användare.

Denna sårbarhet är vanligt förekommande och kan ha allvarliga konsekvenser om den utnyttjas av en angripare. I detta teorikapitel kommer vi att beskriva vad Reflekterad XSS är, hur det fungerar och vilka tekniker som kan användas för att identifiera och åtgärda det. [4]

2.3 Hur fungerar Reflekterad XSS

Reflekterad XSS fungerar genom att en attackerare injicerar skadlig kod in i en webbapplikation som sedan körs på offrets webbläsare. Detta sker vanligtvis via användandet av input validation tekniken. Webbapplikationen kontrollerar eller validerar inte data som användare skickar in. Detta gör det möjligt för en angripare att injicera skadlig kod i applikationen som sedan visas på hemsidan i form av sökresultat eller från en annan källa som via ett skickat Email eller en annan hemsida. När en användare blir lurad

att klicka på en skadlig länk, injiceras och exekveras skadlig kod.

Detta epostmeddelande ska se så äkta och imitera den säkra hemsidan så mycket som möjligt. En annan approach är att skicka en verifieringslänk till tex inloggning hos instagram. Där instagrams riktiga länk har strukturen: *instagram.com/?verifyrequest=true* medans attackerarens länk *instagram.com/?veriifyrequest=true*. Attackeraren stoppar in tex in ett extra *i* för att göra länken snarlik och enkel att missa. Men trycker offer på attackerarens länk exekveras den skadliga koden i stället. [4]

2.3.1 Inputvalidering

Inputvalidering är en teknik som används för att filtrera användarinput i webbapplikationer. Genom att validera användarinput kan en applikation förhindra att illvillig kod injiceras i diverse inmatningsfält som sedan reflekteras i sidans output eller visas för användare via andra metoder. En brist på skydd mot inputvalidering möjliggör därmed attacker i form av reflekterad XSS.

Inputvalidering handlar om att filtrera specialtecken som tex `<`, `>` och `"`. Hanteras inte dessa tecken kan attackerare injicera HTML- eller JavaScript-kod. Det är denna kod som är attackerare skapar som är skadlig. En annan brist i inputvalidering är inom dataformat, om till webbapplikationen inte validerar dataformat kan attackerare injicera SQL-kod [5]

2.3.2 Outputfiltrering

Outputfiltrering är en teknik som används för att filtrera en hemsidas output. Syftet med outputfiltrering är att ytterligare säkerställa att inga skadliga script eller annan potentiellt skadlig kod kan inkluderas och exekveras i det aktuella HTML-dokumentet. Viktig detalj med outputfiltrering är att den inte skyddar mot injektioner som SQL-injektion och Server Side injektion.

Likt inputvalidering handlar det om att identifiera och filtrera bort specialtecken som skulle kunna göra det möjligt att HTML, JavaScript eller SQL-kod exekveras. [5]

3 Metod

För att lösa problemet med reflekterad XSS-sårbarheter inom en webbapplikation krävs en kombination av olika metoder och tekniker. Detta kapitel kommer beskriva de metoder som använts för att lösa uppgiften.

3.1 Projektprocess

Projektet kommer utföras med följande steg:

- Bakgrund och problemformulering
- Teori
- Avgränsningar
- Implementation

3.1.1 Bakgrundsstudie

Bakgrund till sårbarheter på webbapplikationer utforskas och etablerar en stabil grund till vikten av utförandet av projektet generell vetskap till diverse sårbarheter. Mycket material kring bakgrund till sårbarheter finns på OWASPs hemsida.

3.1.2 Identifiera XSS-sårbarheter

Projektet inleds med en teoretisk studie kring olika sårbarheter för webbapplikationer. Därpå Cross-Site scripting valdes med inriktning Reflekterad XSS. Sårbarheter och generell information kring reflekterad XSS studerades från publikationer från OWASP.

3.1.3 Avgränsningar

Varpå projektets lösning endast ska simulera *hur* en attack går till kommer inte någon skadlig kod faktiskt exekvera. Däremot kommer alla steg som leder upp till att en användare navigerar till den farliga koden demonstreras i stället. Dessutom brukar reflekterad XSS använda sig av snarlika utseenden som den trovärdiga sidan skulle se ut. Detta kommer inte heller demonstrationen skapad för projektet att göra.

3.1.4 Implementering av XSS-sårbara hemsidan

Då den specifika sårbarheten är vald påbörjades implementeringen. En hemsida skapas med XSS-sårbarheter vilket automatiskt distribuerar en länk till den angivna E-mail adressen. E-postmeddelandet innehåller en länk där skadligt script exekveras.

4 Konstruktion

Bakgrundsstudien utfördes först kring generella sårbarheter på webbapplikationer. Det första viktiga beslutet i projektet handlade om exakt vilken sårbarhet projektet skulle specificera och demonstrera. Där XSS valdes och fokuserade på den reflektiva varianten.

För att undersöka och demonstrera ett exempel på en reflektiv XSS attack skapades en simpel hemsida. När en hemsida ska skapas är det viktigt att rätt programspråk väljs för att göra det möjligt att utföra alla funktionaliteter som krävs.

HTML används för att skapa en simpel grundstruktur. Strukturen skapar och namnger formulär och textinmatning. Detta gör det möjligt att formatera om hemsidan till ett mer tillfredställande utseende.

CSS används som sagt för att snygga till sidan och göra den mer lättläslig men också enklare att hantera.

PHP används för det mesta i hemsidan. Den skapar och bibehåller Sessionsvariabler, avläser POST-förfrågningar, jämför lösenord, omdirigerar användaren till andra sidor och skickar slutligen ett automatiskt genererat E-postmeddelande.

Konstruktionen av projektet låg i att skapa en enkel PHP-hemsida som kommer ha en väldigt simpel struktur och funktionalitet. Hemsidan kommer endast visa ett inloggningsformulär där den kommer ta emot två användarinputs. Den första inputen är en giltig E-mailadress och den andra är ett lösenord. För enkelhetens skull kommer lösenordet hårdkodad och eftersom detta endast ska vara en demonstration på hur data kan reflekteras.

Många olika informationskällor har använts och granskats för att försöka sammanställa information som är så faktisk som möjligt.

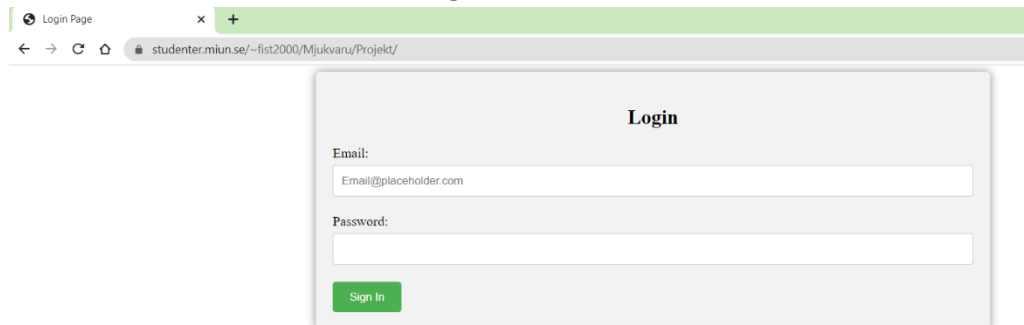
Konstruktionsavsnitt ingår ofta i tekniska rapporter, men inte alltid i vetenskapliga rapporter. Här genomför du din analys av problemställningen och formulerar en teknisk kravspecifikation. Här beskriver du de viktigaste principerna i de lösningsalternativ som du föreslår, utformar och senare i rapporten kommer att utvärdera. Beskrivningen placeras ibland före, men oftast efter metod- /modellkapitlet.

Tänk på att läsaren sällan är intresserad av alltför detaljerad dokumentation av datorprogramkod, algoritmer, kretsscheman, användarhandledning, med mera. Sådana detaljer placeras med fördel i bilagor, om de över huvud taget inkluderas.

5 Resultat

Den resulterade hemsidan slutade som figur 2 till 7.

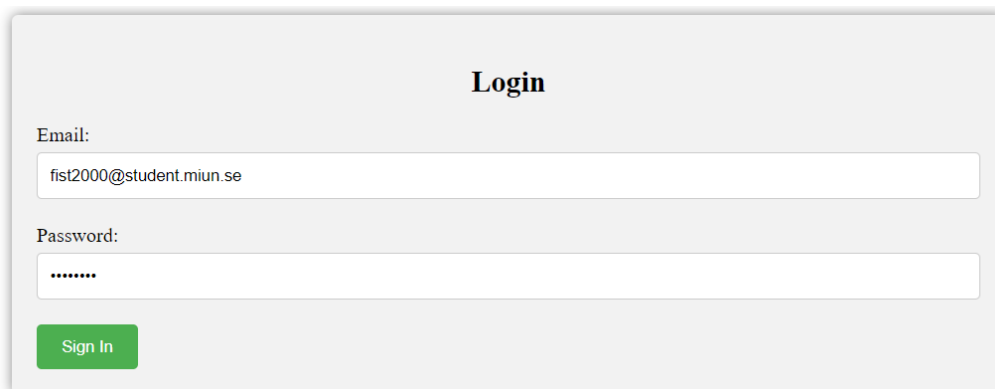
Användaren anländer först till figur 2 där de sedan matar in deras Email-



Figur 2, Steg 1 av resultatet.

adress och det hårdkodade lösenordet som demonstreras i figur 3. För denna demonstration av experimentet används min privata studentmejl och lösenordet *password*:

Figur 3 demonstrerar hur en exempelinput ser ut från en användare



Figur 3, steg 2 av resultatet

När användaren trycker på Sign in slussas de vidare till en annan sida; success.php, här uppmanas användaren att navigera sig till sin angivna email och fortsätta därifrån.

Detta är informationssidan som säger att det lyckades och att ett mejl har skickats till den angivna mailen, ett exempel på utseendet och output ges i figur 4.



Please confirm that you want to login, fist2000@student.miun.se!

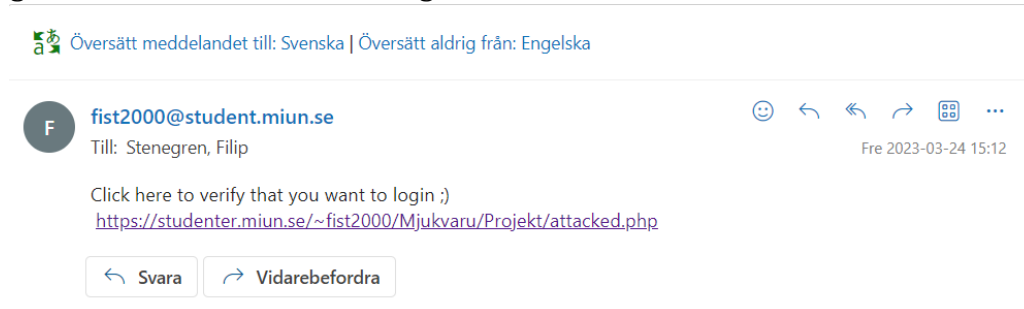
A LINK HAS been sent to your email, follow the instructions there

Your password is: password

Figur 4, steg 3 av resultatet

Sidan uppmanar användaren att fortsätta ifrån deras email.

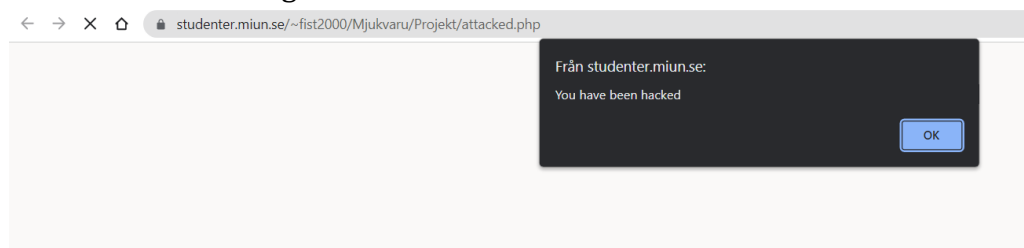
När användaren väl navigerat sig till den angivna E-mailen får de ett auto genererat meddelandet som figur 5 visar:



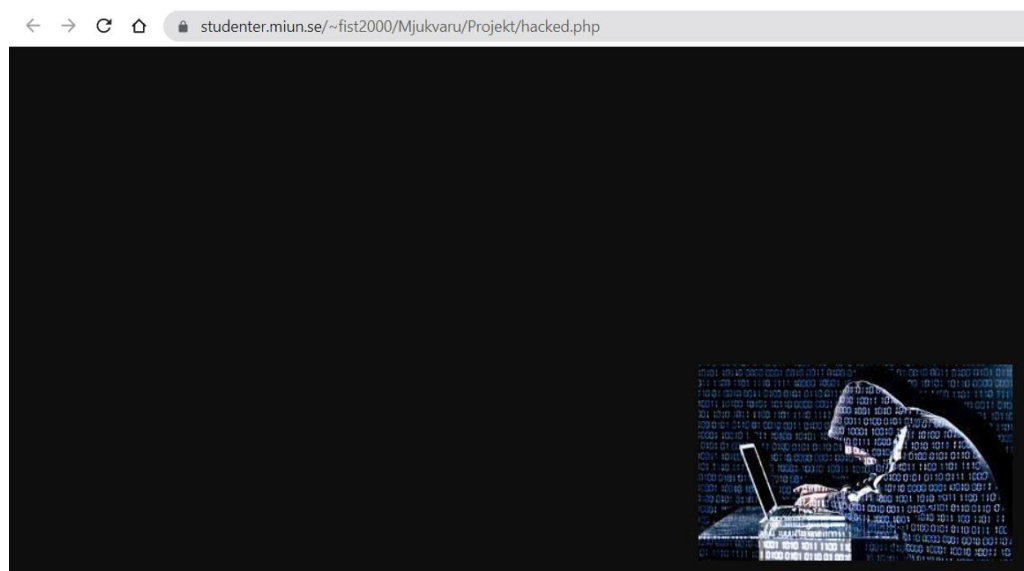
Figur 5, steg 4 av resultatet

Meddelandet förklarar vidare att användaren ska trycka på länken för att *verifiera* att de ska logga in på sidan, och lurar användaren att navigera till den skadliga webbplatsen.

Om användaren trycker på länken skickas de till en sida där den skadliga koden förvaras. När användaren har klickat vidare på länken är det redan för sent. Detta ska figurerna 6 och 7 demonstrera.



figur 6, steg 5 av resultatet



figur 7, steg 6 av resultatet

Om användaren trycker på länken med den skadliga koden har en reflektiv XSS attack lyckats!

6 Slutsatser

En av de viktigast slutsatserna man kan dra från vår undersökning och demonstration är att webbapplikationers sårbarheter är ett allvarligt ämne. XSS kan ge attackerare möjlighet att injicera skadlig kod och få tillgång till känslig information som tex lösenord och cookies. Genom att utnyttja sårbarheten kan angripare till och med manipulera en hemsidas innehåll

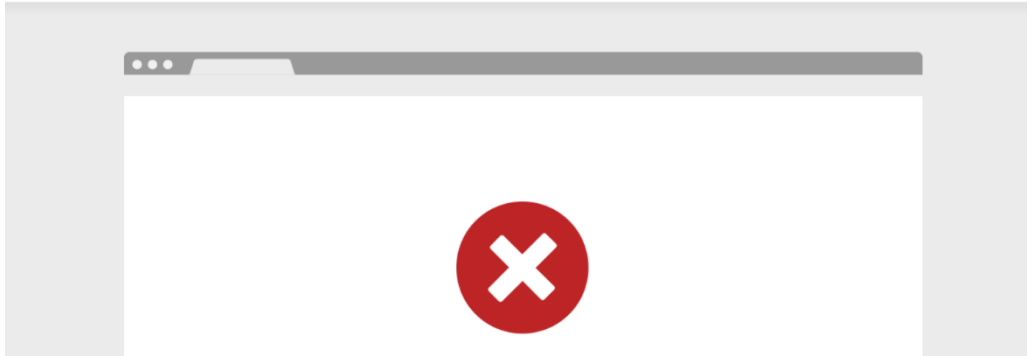
Under projektets gång identifierades flera brister i vår undersökning som kan ha påverkat resultatet. En av de större bristerna var att jag endast undersökte specifik kod för att demonstera sårbarheten. Detta betyder att jag inte kan generalisera eller applicera resultaten till andra webbapplikationer. Sen finns även risken att det finns andra sårbarheter i samma kod som jag inte upptäckte. En annan brist är att jag endast specificerade mig på reflektiv XSS och anpassade inte en mer generell approach där XSS är centralt och reflektiv XSS därav blir inkluderad.

För att skydda mot reflekterad XSS rekommenderar jag att använda en kombination av manuella tester men också verktyg som automatiskt skannar av en hemsidas sårbarheter. Utöver detta rekommenderar jag att inte bara skydda mot XSS utan mot så många internetattacker som möjligt för att försäkra sig att sin webbplats kan bli betrodd.

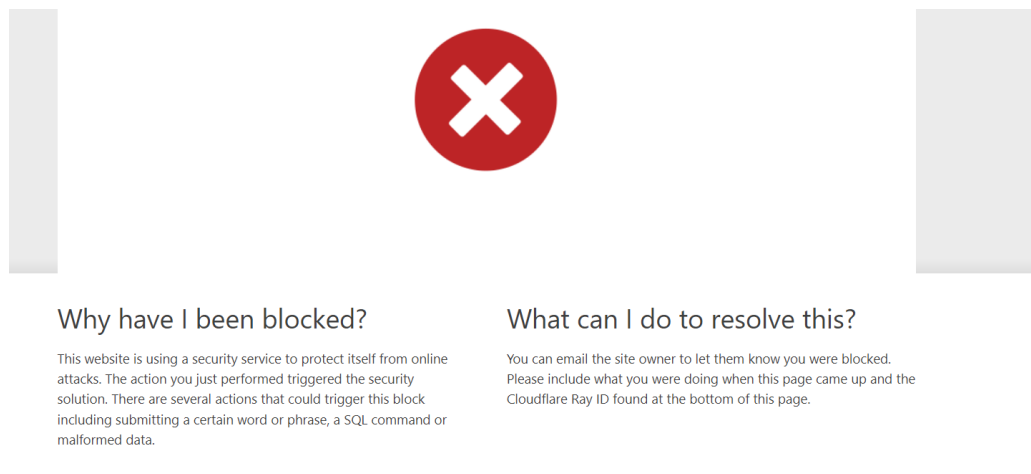
Vid min teoretiska studie påkom jag många sidor som förklara enkla tester för att undersöka om en sida är sårbar för XSS. Varpå jag provade ett exempel på en sida jag tänkt använda som källa men som smart nog aktiviserat skydd mot internetattacker. Den attack jag testade var att skriva in `<script>alert("XSS")</script>` i ett sökfält. Jag vart snabbt omdirigerad till sidan som figur 8 och 9 visar.

Sorry, you have been blocked

You are unable to access www.veracode.com



Figur 8, Exempel på skydd mot internetattacker



Figur 9, Exempel på skydd mot internetattacker med mer info

Detta experiment som jag utförde bevisar även vikten med hur viktigt det är att hemsidor försäkrar och skyddar sig mot inte bara XSS utan även internetattacker generellt men även att många hemsidor just nu har skydd för attacker.

Källförteckning

- [1] OWASP, "Topp 10 lista om säkerhetsrisker 2017",
[https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_\(XSS\)](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS))
Hämtad 2022-03-24
- [2] Veracode, "Säkerhet och information kring XSS",
<https://www.veracode.com/security/xss>
Hämtad 2023-03-20. Blockad 2023-03-24
- [3] Wikipedia, "Cross-Site Scripting"
https://en.wikipedia.org/wiki/Cross-site_scripting
Hämtad 2022-03-24
- [4] OWASP, "Information kring XSS"
<https://owasp.org/www-community/attacks/xss/>
Hämtad 2022-03-24
- [5] PortSwigger, "Cross-Site Scripting"
<https://portswigger.net/web-security/cross-site-scripting>
Hämtad 2022-03-24