Cross-site scripting

XSS

XSS is een type code-injectie-aanval. Het treedt op wanneer een aanvaller misbruik maakt van gaten of kwetsbaarheden in uw webtoepassing, waardoor hij de beveiliging die normaal door browsers wordt opgelegd, kan omzeilen. De aanvaller injecteert een kwaadaardig client-side script in uw webpagina om toegang te krijgen tot gevoelige pagina-inhoud, sessiecookies of andere gebruikersinformatie die door de site is opgeslagen. XSS kan ook worden gebruikt om toegangscontrole te omzeilen, zoals het same origin policy (dat eerder werd besproken).

Zo kan de aanvaller het script gebruiken om zijn schadelijke toepassing uit te voeren vanaf een niet-verwante, kwetsbare site die door de gebruiker wordt vertrouwd – vandaar de term cross-site scripting.

Afhankelijk van de gevoeligheid van de gegevens die gebruikers invoeren, kunnen cross-site scripting-aanvallen variëren van een kleine overlast tot een aanzienlijk veiligheidsrisico. Ze zijn populair bij hackers en omvatten de meeste gemelde beveiligingsproblemen.

Hoe XSS werkt

Een XSS-applicatie verzamelt gegevens van de gebruiker, meestal via een hyperlink waar de gebruiker toe wordt verleid om op te klikken, die de schadelijke inhoud bevat. Aanvallers maskeren over het algemeen het kwaadaardige deel van de link met een hexadecimale of andere codering, zodat de link er minder verdacht uitziet.

Nadat de XSS-applicatie de gebruikersgegevens heeft verzameld, genereert deze een uitvoerpagina voor de gebruiker met de gegevens die oorspronkelijk naar de gebruiker zijn verzonden, maar op een manier waardoor het lijkt alsof het geldige inhoud van de site is. Omdat blogs en fora gebruikers in staat kunnen stellen berichten in te sturen met daarin ingesloten HTML- en JavaScript-codering, zijn dit populaire doelen voor XSS.

De invoer van de gebruiker wordt gemanipuleerd en teruggekaatst, wat misschien niet zo'n groot probleem lijkt. Als een aanvaller echter misleidende links of verborgen frames op niet-gerelateerde sites plaatst, kan dit ervoor zorgen dat de browser van de gebruiker naar URL's op de kwetsbare site verwijst – vaak onopgemerkt – waardoor de aanvaller de beveiligingscontext van de gebruiker kan schenden.

In wezen, als de aanvaller een gebruiker ertoe kan brengen om op een link te klikken – op de vertrouwde (maar kwetsbare) site of in een vervalste e-mail die naar de vertrouwde site verwijst – wordt het kwaadaardige script in de link-URL uitgevoerd in de browser van de gebruiker, lijkend op de verwachte inhoud van de vertrouwde site. Dit kwaadaardige script kan een sessiecookie van de gebruiker naar de aanvaller sturen zonder medeweten van de gebruiker. Afhankelijk van de aard van de vertrouwde site kan de cookie gevoelige gegevens van de gebruiker bevatten, zoals een wachtwoord of factuurgegevens. Een cookie kan ook worden gebruikt om een gebruikerssessie op een forumsite te kapen, zodat de aanvaller zich op die site kan voordoen als de gebruiker. Later leert u meer over cookies.

Soorten XSS

Er zijn drie algemene typen XSS, die worden beschreven in de tabel:

XSS Type	Description	Example
Reflected (Non-persistent)	The most common type. These holes occur when data provided by a browser is used immediately by server-side scripts to generate a page of results for that user. If invalidated user-supplied data is included in the resulting page without HTML escaping (entities for special characters), then the client-side code can be injected into the dynamic page.	A user performs a site search for a string that includes HTML special characters, and the search string is redisplayed on the result page. If the search terms are not encoded as HTML entities, an XSS hole will result.
Persistent	Enables the most powerful attacks. When a user enters data in the browser, it is first stored persistently on the server (in a database or file system, for example), then later displayed to other users in a webpage without proper HTML escaping.	Some online message boards allow users to post HTML-formatted messages for other users to read. An attacker can obtain a session cookie and impersonate the user in the forum.
DOM-Based	This problem exists in a page's client-side script itself. A JavaScript program accesses a URL request parameter and uses it to write some HTML to its own page. If this information is not encoded using HTML entities (escaping), it can create an XSS hole because the written information will be re-interpreted by browsers as HTML, which might include additional client-side script. With Internet Explorer, which runs a client-side script in objects located in the local zone (e.g., the client's local hard drive), an XSS hole of this type in a local page can result in remote execution vulnerabilities.	An attacker hosts a malicious site with a link to a vulnerable page on a client's local system. The hacker could inject a script that would run on the user's system with the privileges of the browser. This bypasses the entire client-side sandbox, not just the cross-domain restrictions that are normally bypassed with XSS exploits.

Code en servers in XSS

XSS is in wezen een code-injectie-aanval. De code van de aanvaller stuurt kwaadwillende gebruikersinvoer van de server naar de browser van het slachtoffer, allemaal binnen het HTTP-antwoord dat de browser ontvangt om op het scherm weer te geven voor de gebruiker. De browser kan geen onderscheid maken tussen de echte site-inhoud en gebruikersinvoer die mogelijk schadelijk is, dus de browser geeft de kwaadaardige code weer als HTML of een scriptopdracht, in plaats van de platte-tekstreeks die het had moeten zijn. Het resultaat is een gat of kwetsbaarheid in de site.

Servers hebben doorgaans een beleid dat voorkomt dat scripts of database-informatie van de ene site op een andere site wordt uitgevoerd. Maar XSS omzeilt deze beperking, waardoor een hacker veel sites vanaf één site kan hacken.

XSS kan ook in uw e-mail verschijnen. Een gebruiker kan e-mail ontvangen van wat een vertrouwde website lijkt te zijn. Door op een link in het e-mailbericht te klikken, heeft de gebruiker mogelijk inderdaad toegang tot de vertrouwde site (zoals een bank), maar is schadelijke code in de URL ingesloten, waardoor de hacker wachtwoorden of bankgegevens kan stelen die de gebruiker verzendt tijdens die sessie.

Wat JavaScript-ontwikkelaars kunnen doen

Over het algemeen moet u de invoer van gebruikers niet vertrouwen en moet u de uitvoer altijd coderen om metatekens te filteren. Dit helpt de meeste XSS-aanvallen te voorkomen.

Output encoding

Output encoding is het beste verdedigingsmechanisme voor XSS. Dit betekent dat bepaalde speciale tekens correct als HTML-entiteiten moeten worden gecodeerd. Uitvoercodering verzekert u dat informatie die van de server terugkeert, opnieuw wordt verwerkt en naar HTML omgezet, zodat wanneer de browser de informatie ontvangt deze als HTML wordt getoond en niet als code uitgevoerd.

Invoer validatie

Alleen uitvoercodering is niet voldoende omdat veel toepassingen gebruikersinvoer in HTML-formaat accepteren om de invoer in de browser weer te geven. Inputvalidatie kan helpen. Het valideren van niet-vertrouwde HTML-invoer is echter een complexe taak waarvoor een HTML-beleidsengine vereist is die op XSS kan controleren. Hiervoor zijn tools van vertrouwde beveiligingsorganisaties zoals OWASP beschikbaar.

Invoervalidatie kan echter niet alle XSS eruit filteren. Een van de redenen is dat de verscheidenheid aan toepassingen op het web tegenwoordig validatie bemoeilijkt. Het mashup-model betekent bijvoorbeeld dat u soms de bron van een applicatie niet weet of dat het validatie gebruikt, volledig of helemaal niet. Een andere reden is dat 'zero-day exploits' direct kunnen profiteren van een situatie, vaak nog voordat de antivirussoftware van het systeem kan reageren en een patch voor het virus kan bieden.

Cookies en XSS

Omdat sessiecookies vaak het doelwit zijn van XSS-aanvallen, kunt u proberen cookies te beveiligen door sessiecookies te koppelen aan het IP-adres van de gebruiker die zich oorspronkelijk heeft aangemeld, en vervolgens alleen dat IP-adres toe te staan om die gerelateerde cookie te gebruiken. Deze praktijk kan nuttig zijn wanneer de cookie het doelwit is, hoewel een ervaren hacker ook het IP-adres kan vervalsen.

Scriptblokkering en XSS

Waarschijnlijk is de meest effectieve manier voor gebruikers om te voorkomen dat ze worden aangevallen door een script, te voorkomen dat scripts in hun browser worden uitgevoerd. Het blokkeren van alle scripts betekent echter minder functionaliteit en reactievermogen en een aanzienlijke verslechtering van de surfervaring op het web. Net als al het andere moet er een prijs worden betaald voor alles wat u van internet krijgt, en die prijs is het risico van kwetsbaarheid.

OWASP en XSS

Het Open Web Application Security Project (OWASP) is bedoeld om ontwikkelaars te helpen hun sites te beveiligen. Voor meer informatie over het voorkomen van XSS kunt u de XSS Prevention Cheat Sheet bezoeken op de volgende pagina: Cross-Site Scripting - Cheatsheet voor preventie.