Cross-site Scripting (skraćeno *XSS*) je napad izvršen sa klijentske strane. Napadač pokušava pokrenuti maliciozne skripte u web browseru žrtve tako što maliciozni kod ubaci u pouzdanu i legitimnu web stranicu ili aplikaciju. Napadnuta web stranica ili aplikacija postaje prenosno sredstvo za skriptu, te je dovodi do browsera.

Česte situacije gdje napadači pokušavaju izvesti XSS su forumi, message boards i razne web stranice koje dozvoljavaju javno ostavljanje komentara.

XSS napadi su mogući u VBScript, ActiveX, Flash, te čak i u CSS-u, međutim, daleko *najčešći* su u vidu JavaScripta. Razlog za ovo leži u činjenici da je JavaScript fundamentalan većini današnjih browsera.

Šta tačno napadač može postići pomoću XSS napada?

* promijeniti sadržaj napadnute web stranice
* preusmjeriti korisnika na drugu stranicu (često takođe malicioznu)
* maliciozni JavaScript može pristupiti svim objektima kojima pristupa i web stranica, *uključujući user cookies*. Cookies sadrže informacije o sesiji svakog korisnika, te ukoliko se napadač uspije dokopati cookie-ja, može impersonirati korisnika čiji je cookie ukrao, te izvršavati sve akcije za koje taj korisnik ima privilegije.
* JavaScript u modernim browserima može pomoću HTML5 API pristupiti korisnikovoj geografskoj lokaciji, webcam-u, mikrofonu i čak nekim fajlovima iz korisnikovog file sistema.
* Moguće je kompromizovanom korisniku ubaciti trojanca u sistem.
* Moguće su krade identiteta.

Postoji nekoliko tipova XSS napada:

* stored/persistent XSS
* reflected/non-persistent XSS
* DOM-based XSS

Kako funkcionišu XSS napadi?

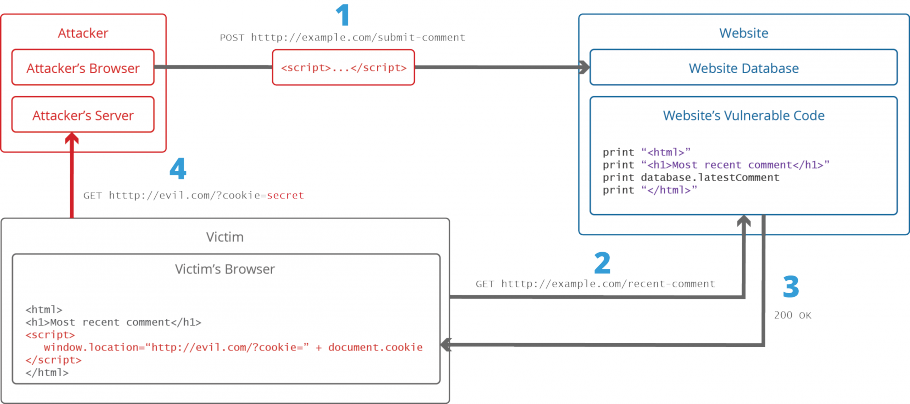
Svaki tip XSS napada se sastoji iz dva osnovna koraka:

1. Pokrenuti maliciozni JS kod u korisnikovom browseru. Kako bi napadač ovo ostvario, on mora prvo pronaći način da ubaci neki malicioznu skriptu na neku web stranicu koju taj korisnik posjećuje.
2. Nakon što se to uradi, čeka se da korisnik pristupi stranici na kojoj se sada nalazi maliciozna skripta. Ako napadač *tačno* zna ko mu je meta, može koristiti *phishing[[1]](#footnote-1)* da pošalje maliciozni URL korisniku. Zbog prirode *phishing*-a, taj URL će korisnikovom browseru izgledati kao da je došao iz provjerenog izvora, iako nije.

Kako bi se prvi korak mogao ostvariti, potrebno je da stranica koju napadač napada direktno koristi input od korisnika u svom radu. Napadač onda unosi maliciozni string koji će biti korišten u web stranici, te koji će browser ugroženog korisnika tretirati kao *source code* te stranice.

Kada se stranica učita u klijentovom browseru, napadačeva skripta se izvršava i klijent je ne može zaustaviti.

Krađa cookies



1. Napadač ubaci *payload[[2]](#footnote-2)* u bazu tako što u ranjivu input formu ubaci maliciozni JS kod.
2. Klijent (ne znajući) pošalje zahtjev web serveru da mu dobavi tu stranicu.
3. Web server opsluži klijentov browser tako što mu pošalje unesenu malicioznu skriptu kao dio HTML body.
4. Klijentov browser izvršava malicioznu skriptu koja se nalazi u tijelu HTML dokumenta. U ovom slučaju browser šalje *cookie* napadačevom serveru.
5. Sve što napadač sada mora da uradi jeste da izvuče (eng. *extract*) klijentov *cookie* kada mu HTTP request stigne na server.
6. Napadač može koristiti ukradeni cookie za impersonaciju.

Najčešći XSS vektori[[3]](#footnote-3)

<script> tag

Ubjedljivo najčešći i najdirektniji XSS payload. Pomoću <script> taga referenciramo neki eksterni JS kod ili ga sami pišemo.

JS events

onload i onerror events se mogu koristiti unutar raznih tagova, što ih čini itekako popularnim XSS payloadom.

<body> tag

U <body> tagu napdač može pomoću JS events ili atributa body taga da ubaci maliciozni kod. Može koristiti atribute koji sami po sebi nemaju nikakve veze sa samim JavaScriptom, kao npr background.

<img> i <input> tag

Neki browseri su u stanju izvršiti JS kod koji se nalazi u src atributu <img> i <input> tagova koji za tip imaju type=”image”.

<iframe> tag

<iframe> tag omogućava da se jedna HTML stranica “ugradi” (eng. *embed*) u stranicu koja se trenutno prikazuje klijentu. Oni se konkretno koriste za *phishing*, odnosno za namamljivanje korisnika na neki maliciozni URL.

<link> tag

Ovaj tag se inače koristi za referenciranje eksternih skripti, te se kao takav može koristiti da se

<table> i <div> tag

background atribut <table> i <div> tagova se može iskoristiti da se referencira skripta umjesto neke slike.

<object> tag

Ovaj tag se može iskoristiti za referenciranje eksterne skripte postavljajući data atribut na URL pomenute skripte.

Više o tipovima XSS

Stored XSS

Ovo je tip XSS napada koji je u stanju nanijeti najveću štetu klijentu koji je meta napdaču. Napadač će iskoristiti ovaj tip napada da ubaci maliciozni kod (najčešće pisan u JavaScriptu) u ranjivu web aplikaciju. Ako nismo osigurali validaciju podataka koje korisnici unose u naše HTML forme, ovaj kod se *direktno* unosi u neki vid trajnog čuvanja podataka, kao na primjer u bazu.

Česte situacije u kojima se pojavljuje su blog postovi i forumi.

Reflected XSS

Drugi najčešći tip XSS napada. U ovom slučaju, napadačev *payload* mora biti dio zahtjeva koji se šalje web serveru. Nakon ovoga se vraća nazad (end. *is reflected*) na način da HTTP response sadrži payload koji se je nalazio u requestu. Napadači često podmuklo koriste metode poput *phishing*, maliciozne linkove i sumnjive tehnike da namame klijenta da neznajući pošalje zahtjev serveru.

Ovi tipovi napada najčesšće imaju jednu konkretnu žrtvu koju žele da sabotiraju, a često se javljaju na društvenim mrežama.

DOM-based XSS

Ovo je najnapredniji i najkompleksniji tip XSS napada. Ono što ga omogućava jeste situacija u kojoj skripte web aplikacije koje rade na client-side zapisuju podatke (koje im korisnik daje u DOM (Document Object Model).

U ovom slučaju, payload se ne šalje na server uopšte, te ga zato WAFs (skraćeno od *Web Application Firewalls*) teško mogu detektovati. DOM objekti koji su najranjiviji su document.URL, location.hash i document.referrer.

Zaštita od XSS napada

U suštini, svi XSS napadi nastoje da napadnu neki web site pomoću inputa koje daje korisnik. Kod koji će napasti stranicu može varirati u svojoj kompleksnosti od jednostavne input forme, do komplikovanih JSON skripti, ovisno od tog šta napadač tačno želi da postigne. Glavna ideja zaštite je dakle, da je developer u svakom momentu svjesan da mu podaci dolaze iz vanjskih izvora kojima ne treba slijepo vjerovati.

Shodno tome, nameće se nekoliko metoda zaštite od XSS napada.

Filtriranje

Najjednostavnija ideja jeste da se sav input koji se uzima od korisnika provuče kroz neki vid filtera koji će analizirati i ukloniti sve potencijalno opasne ključne riječi, poput <script> taga, JS komandi, CSS stilova (koji takođe mogu biti maliciozni) i generalno svih HTML markupa koji sadrže event handlers.

Mnogi web developeri sami razvijaju vlastite filtere na server-side. Najčešće se razvijaju u PHP-u, mada se mogu koristiti i drugi jezici. Filter traži ključne riječi koje mu zada developer i mijenja ih praznim stringovima.

Moglo bi se pomisliti da su regularni izrazi dobra ideja za podlogu filtera, međutim pokazalo se da u praksi to nije tako. Razlog leži u činjenici da su napadači u većini slučajeva puno iskusniji od developera, te da ovako jednostavne filtere (kao te koji su bazirani na regularnim izrazima) mogu bez većih poteškoća zaobići. Iz tog razloga se preporučuje da se koristi neka temeljito istestirana biblioteka.

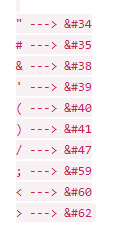
Biblioteka koje se koriste u ovu svrhu ima mnogo, a ono što određuje izbor biblioteke jeste *backend* tehnologija koja je korištena za razvoj aplikacije. Uvijek nam je u interesu da odaberemo onu biblioteku koja je ažurna i koja stalno updateuje svoje sigurnosne metode i tehnike.

Jedina mana ovog pristupa je što će prestrog filter zabraniti unos nekih legitimnih riječi, pa se o tome treba povesti računa.

Escaping

*Escaping* je tehnika pomoću koje klijent kaže browseru da sve što mu se pošalje kao odgovor treba da se tretira kao podatak, i ni na kakav drugi način. Ono što se ovom metodom postiže je to da browser *ne* izvršava skriptu koju je napadač ubacio u stranicu koju smo zatražili.

U HTML-u je escaping relativno jednostavan za izvesti. Sve što treba da uradimo jeste da opasne simbole zamijenimo njihovim escape code-om i naš posao je gotov.

  
Escape kodovi za “opasne” simbole

Ovo će browseru efektivno reći da se ovi simboli *ne trebaju* izvršavati, već samo parsirati kao najobičniji tekst, što napadačevu skriptu čini uzaludnom.

Nešto teže je uraditi escaping za JS, CSS i XML, i nažalost, to je upravo ono što je neophodno da se uradi kako bi naša zaštita bila potpuna. Veoma je teško ručno uraditi escaping za sve tri navedene tehnologije, i upravo se iz tog razloga uvijek preporučuje da se koriste isprobane escaping biblioteke kao ESAPI (omogućio OWASP) i AntiXSS (omogućio MiscroSoft).

Još jedan izazov postoji s ovom metodom, a to je kako znati koliko escaping-a je previše escaping-a? Naime, problem se javlja u tome što nikada ne možemo uraditi escaping *za sve*, jer u tom slučaju naša stranica više neće biti funkcionalna - sav JS kod bi bio escapiran, kao i HTML i CSS.

Postoje određene preporuke kako bi se izbjegao ovaj problem:

* koristiti HTML escaping kada se sumnjiv kod ubaci između početnih i završnih HTML tagova
* koristiti JavaScript escaping kada se sumnjivi podaci ubace u jednu od skripti ili na neko drugo mjesto gdje se JS kod može neometano izvršiti. Ovo uključuje neke atrivute kao što je style i sve event handlers.
* koristiti CSS escaping kada se sumnjivi podaci ubace u jedan od CSS stilova. Kao što smo vidjeli ranije, mnogi CSS atributi se mogu zloupotrijebiti u ovu svrhu.

NAPOMENA

Na stranici referenciranoj ispod (prva po redu) imaš screenshote svakog payloada. Ne bi bilo loše da ih sama otkucaš i probaš pokrenuti and all that da vidiš hoće li ti se ukazati ta skripta or whatever the hell it is. U suprotnom, copy paste i dosta.

Reference

<https://www.acunetix.com/websitesecurity/cross-site-scripting/>

<https://www.acunetix.com/blog/articles/preventing-xss-attacks/>

<https://www.acunetix.com/websitesecurity/xss/>

1. *Phishing* je napad krađe prilikom kojeg se napadač pretvara da je pouzdan izvor (kao recimo, sigurna i validirana web stranica), te nastoji ukrasti usernames, šifre i ostale osjetljive informacije o korisniku kako bi ih kasnije koristio u vlastite svrhe. [↑](#footnote-ref-1)
2. Sinonim za maliciozni kod. [↑](#footnote-ref-2)
3. Način ubacivanja malicioznog koda. [↑](#footnote-ref-3)