CARNet

HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

CSRF napadi

NCERT-PUBDOC-2010-04-297

Nacionalni CERT

u suradnji s

LSS

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem Nacionalni CERT kontinuirano radi.

Rezultat toga rada je i ovaj dokument, koji je nastao suradnjom Nacionalnog CERT–a i LS&S–a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

Nacionalni CERT, www.cert.hr

Nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, www.LSS.hr

Laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Sadržaj

1. Uvod

Sigurnost *web* programa danas je jedno od ključnih pitanja u računalnoj sigurnosti. Riječ je o programima koji se s udaljenog poslužitelja učitavaju u *web* preglednik klijenta. Često se zasnivaju na distribuiranoj klijent/poslužitelj arhitekturi, a danas se koriste kod velikog broja Internet usluga, od društvenih mreža do bankovnih mrežnih sjedišta. Zbog raširenosti ovakvog načina rada, potencijalnih koristi za napadače, ali i privlačnosti za hakere koji se žele dokazati (npr. napadima na društvene mreže i korisničke profile), napadi na *web* programe i *web* stranice iznimno su česti.

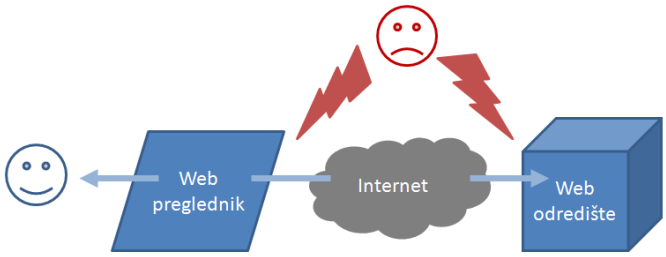
Raspon različitih metoda koje se koriste pri narušavanju sigurnosti *web* sjedišta je iznimno širok, a uglavnom se svodi na nedovoljnu provjeru sadržaja koji stranice međusobno razmjenjuju i povezanost između sadržaja koji se prikazuje i koda koji upravlja prikazivanjem tog sadržaja. Jedna od metoda koja se oslanja upravo na ovu specifičnu povezanost sadržaja i koda u *web* programima je CSRF (eng. *Cross-Site Request Forgery*). Riječ je o ranjivosti koja proizlazi iz nedovoljne provjere zahtjeva za *web* sadržajem i samog *web* sadržaja koji se učitava. Učitani *web* sadržaj može ujedno sadržavati i zahtjeve za drugim *web* sadržajem te pokrenuti njihovo slanje iz korisnikovog *web* preglednika. Ukoliko se pritom koriste prethodno uspostavljene veze između korisnika i drugog *web* odredišta, zloćudna stranica može svoj podmetnuti zahtjev iskoristiti za izvođenje radnji u ime korisnika. Naime, ako je uspostavljena veza s ranjivim odredištem zapamćena u *web* pregledniku, on će prilikom slanja zahtjeva, učitanog s druge stranice, automatski koristiti odgovarajuće autentikacijske kolačiće za ranjivu domenu.

U ovom dokumentu dan je uvod u svojstva CSRF napada i rizike koje izazivaju. Predstavljene su srodne ranjivosti s kojima se zajedno mogu koristiti kako bi se povećala učinkovitost napada. Dani su i primjeri te kratki opisi jednostavnih metoda izvedbe napada HTML objektima ili automatiziranim pokretanjem zahtjeva u *web* preglednicima. Na kraju dokumenta navode se kratki savjeti za zaštitu od ove vrste napada sa strane klijenta i poslužitelja.

2. Sigurnost web aplikacija

Web aplikacijama se pristupa preko računalne mreže - Interneta i preko *web* preglednika unutar kojeg se oni dijelom izvode. Riječ je distribuiranim klijent/poslužitelj sustavima koji se osim na korisnikovom, tj. klijentskom računalu (eng. *client-side*), izvode i na *web* poslužitelju (eng. *server-side*). Pisani su najčešće u kombinaciji jezika kao što su JavaScript i HTML. Primjer klijentskog programa je *web* preglednik. On učitava i pokreće programski kod s udaljenog *web* poslužitelja te podnosi zahtjeve za određenim stranicama i radnjama. To se odvija svakodnevno preko Internet Explorera, Mozille Firefoxa, Safarija, Opere i drugih *web* preglednika. Razmjena *web* sadržaja može biti uzrokovana upisom nove stranice, pretraživanjem Interneta, klikom ili čak samo pomakom miša. To su radnje o kojima korisnici ne razmišljaju dok ih čine, a koje u svom programskom kodu mogu otvarati prostor za zlouporabe.

Jedna od najčešćih meta internetskih napada su upravo *web* preglednici. Napadači iskorištavaju ranjivosti u njima kako bi nanijeli štetu računalu korisnika, ukrali podatke ili napali drugu *web* stranicu preko korisnikovog *web* preglednika. U zaštiti *web* poslužitelja i klijenata, odnosno podataka koje oni razmjenjuju važan je velik broj elemenata koji uključuju odgovornog korisnika, siguran i redovito nadograđivan *web* preglednik, pouzdanu DNS uslugu, te sigurno oblikovane *web* aplikacije. CSRF ranjivost o kojoj se govori u nastavku dokumenta vezana je najčešće uz nesigurno oblikovane *web* poslužitelje te uz nesiguran rad *web* preglednika. Kod *web* poslužitelja rizici se javljaju ukoliko se ne provjerava identitet pošiljatelja zahtjeva niti odgovara li oblik i sadržaj podataka onom koji se očekuje za pojedinu vrstu obrade. Kod *web* preglednika problem je i nedovoljna provjera podataka, ali u ovom slučaju ona se odnosi na mogućnost lažiranja zahtjeva, tj. na postojanje mogućnosti da se u *web* pregledniku stvori i pošalje zahtjev u ime korisnika koji ga nije inicirao.



Slika 1. Web programi i Internet

2.1. CSRF ranjivost

CSRF (eng. *Cross-Site Request Forgery*) je vrsta napada na ranjivu *web* stranicu koja se izvodi posredstvom klijentskog programa. Ukoliko se zloćudna stranica učita u nesiguran *web* preglednik moguće je pokrenuti zahtjev prema drugoj (ranjivoj) stranici u ime korisnika *web* preglednika. Pritom zloćudna stranica mora sadržavati kod za pokretanje takvog zahtjeva koji će se s ostatkom koda te stranice automatski učitati i pokrenuti u *web* pregledniku. Ukoliko ranjiva stranica ne izvodi provjere sigurnosti i identiteta korisnika, zahtjev poslan posredno sa zloćudne stranice obradit će se kao da ga je poslao klijent.

Ranjivost može zahvatiti bilo koji *web* sustav koji ne zadovoljava principe sigurnog oblikovanja i ne provjerava autentičnost primljenih HTTP zahtjeva.

2.1.1. Osobitosti napada

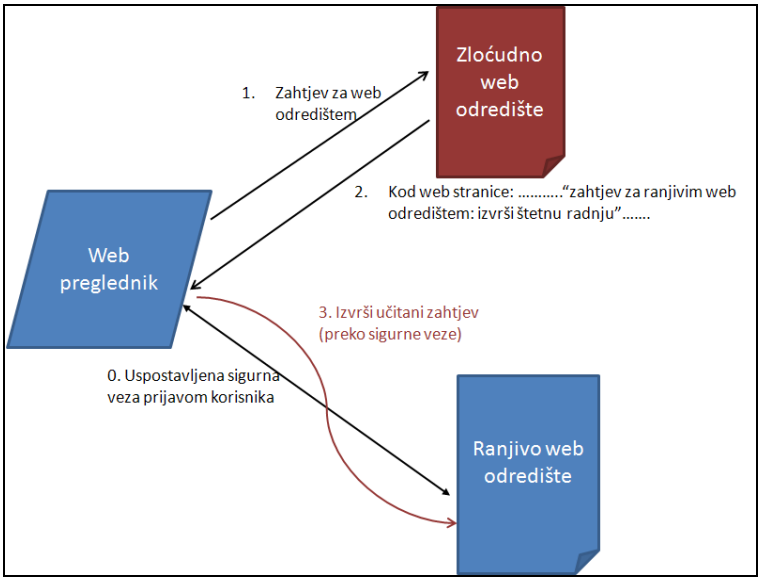
CSRF napad cilja na stranice koje se u interakciji s korisnicima i njihovim aktivnostima oslanjaju na poznavanje korisničkog identiteta. Pritom se iskorištava povjerenje koje je uspostavljeno između korisnika i stranice. CSRF lažira identitet pošiljatelja zahtjeva tako što navodi njegov *web* preglednik na slanje HTTP zahtjeva u ime korisnika. Da bi napad imao učinka koriste se HTTP zahtjevi koji izazivaju određene promjene podataka i sadržaja. Tipični zahtjevi su GET i POST. GET metoda koristi se za dohvat podataka i prenosi se putem URL zahtjeva. Kod POST metode podaci se šalju u tijelu poruke. Ona se koristi za različite transakcije i izmjene podataka. Budući da GET metoda nije predviđena za izmjene već samo za dohvat podataka, često se njezina uporaba u

poslužiteljima ne provjerava dovoljno. Zato postoji mogućnost podmetanja GET zahtjeva koji će mimo sigurnosnih provjera izazvati štetne izmjene podataka na stranici. POST zahtjevi u tom smislu su sigurniji, no svejedno zahtijevaju pažljivu provjeru podataka. Korištenje GET zahtjeva preporuča se jedino u slučajevima kad nema mogućnosti izmjene sadržaja.

CSRF napad se često izvodi preko internetskih foruma koje koristi velik broj korisnika te ih je lako navesti na otvaranje zloćudnih *web* poveznica. One se podmeću kroz slike koje korisnici postavljaju na forum jer postavljanje skripti nije dopušteno.

Preduvjeti koji moraju postojati da bi se mogao izvesti CSRF napad su sljedeći:

- stranica koja se napada ne provjerava izvor poruke, odnosno HTTP Referer zaglavlje ili
- *web* preglednik korisnika omogućuje lažiranje tog zaglavlja,
- koristi se takav HTTP zahtjev koji izvodi promjene na napadnutoj stranici ili korisničkom računu (npr. mijenja lozinku),
- napadač ima pristup autentikacijskim kolačićima i sigurnosnim značkama preko kojih pristupa ranjivoj stranici i
- napadač je u mogućnosti navesti žrtvu na otvaranje zloćudne *web* poveznice.



Slika 2. Shema CSRF napada

Iz preduvjeta se vidi da napad nije tako lako izvediv, osobito zbog potrebe za pristupom autentikacijskim objektima. Također, osobitost ovog napada je da napadač nema uvid u odgovor napadnute stranice na njegov napad jer se on šalje korisniku. Ovaj „problem“ je moguće premostiti pomoću drugih napada o kojima će biti govora kasnije u tekstu (XSS).

2.1.2. Potencijalni rizici

Kao što je navedeno u prethodnom odlomku, CSRF nije tako lako izvesti. O tome govori i činjenica da nije zabilježen velik broj napada ove vrste u odnosu na druge ranjivosti. Ipak problem sa CSRF-om je taj što je doseg njegovih posljedica praktično neograničen. Svaka radnja koja se može na *webu* izvesti pomoću URL poveznice ili predajom *web* obrasca može se izvesti i CSRF napadom. Primjeri takvih radnji su:

- objavljivanje sadržaja na blogovima i forumima u ime korisnika,
- slanje različitih poruka,
- on-line kupovina,
- pretplate na virtualne sadržaje i sl.

Posebno teška okolnost CSRF napada je ta da je u *web* pregledniku sačuvana prijava korisnika na ranjivu stranicu. U tom slučaju *web* preglednik zajedno sa zahtjevom šalje i autentikacijski kolačić s kojim se zloćudan zahtjev na ranjivoj stranici izvodi s ovlastima prijavljenog korisnika. Primjer ovakve zlouporabe može biti napad na internetsko poslovanje banke i korisnički račun određenog

klijenta. Uspješna zlouporaba može dovesti do novčane transakcije s korisnikovog na napadačev račun. Riječ je o vrlo konkretnoj, ozbiljnoj šteti i kriminalnoj radnji. Prijetnje sigurnosti koje postoje na Internetu korisnici često ne doživljavaju ozbiljno. Ovaj primjer pokazuje realnu mogućnost konkretne, ozbiljne štete i kriminala koji korisnik može pretrpjeti zbog nedovoljne brige o sigurnosti (s njegove strane, ali i sa strane pružatelja usluge). Iako valja napomenuti kako je vjerojatnost ovakvog napada zapravo mala jer ozbiljne stranice će nametnuti dodatne provjere identiteta kod svake kritične aktivnosti, a novčana transakcija jedna je od najkritičnijih.

Osim preko *web* preglednika, CSRF se može izvesti i preko drugih programa. To mogu biti programi koji koriste XML dokumente. Oni mogu biti oblikovani tako da dohvaćaju sadržaj s nekog *web* odredišta. Također ranjive mogu biti *Word* i *Flash* datoteke, odnosno svi drugi sadržaji kod kojih postoji mogućnost umetanja skripti i povezivanja na *web*.

2.1.3. Povijest napada i zabilježeni slučajevi

CSRF napad je poznat još od 90-tih godina prošlog stoljeća. Do danas nije zabilježen velik broj slučajeva. Primjerice, izvješće WASC-a (eng. *Web Application Security Consortium*)[11] za 2007. godinu navodi udio od 2% CSRF u svim otkrivenim *web* napadima. Tako mali postotak objašnjava se relativno lakim otkrivanjem ove vrste zlouporabe i malom iskorištenošću. Prema statistici istog konzorcija za 2008. godinu, udio CRFS napada manji je od 2%. Najpoznatiji slučajevi CSRF napada su oni koji su zahvatili vrlo velik broj korisnika. Tako je u napadu na *web* odredište *Auction.co.kr*, korejsko sjedište organizacije *eBay's Internet Auction Co.*, 18 milijuna ljudi izgubilo osobne podatke. Zabilježen je i slučaj napada na meksičku banku u kojem su korisnici preusmjeravani na drugu stranicu te napad na *GMail* koji je neovlašteno otkrivao listu kontakata korisnika. Najpoznatiji napad je bila kombinacija XSS (eng. *Cross-Site Scripting*) i CSRF metode. Riječ je o crvu *Samy* koji se širio *MySpace* društvenom mrežom. Na profilima korisnika ostavljao je poruku *"but most of all, Samy is my hero"*. U roku od 20 sati proširio se na više od milijun profila.

Slika 3. Primjer MySpace profila zaraženog Samy crvom
Izvor: Google Blogscoped

2.2. Napadi srodni CSRF-u

Ranjivosti *web* usluga moguće je iskoristiti na velik broj načina:

- podmetanjem SQL koda kojim se otkrivaju ili mijenjaju podaci u bazi,
- izazivanjem različitih preljeva spremnika koji dovode do DoS (eng. *Denial of Service*) stanja ili do pokretanja proizvoljnog programskog koda,
- preuzimanjem korisničkih sjednica pomoću kolačića i slično.

Primjeri napada koji djeluju na sličan način kao i CSRF napad su XSS i fiksacija ključa korisničke sjednice. Napadi se izvode navođenjem korisnika na učitavanje posebno oblikovanog URL zahtjeva koji u konačnici dovodi do podmetanja lažnog sadržaja korisniku ili do neovlaštenog pristupa drugoj stranici u ime korisnika. Osim što su slični CSRF napadu, mogu se u kombinaciji s njime izvoditi kako bi se povećala njihova učinkovitost. Zbog njihove povezanosti i iskoristivosti u kombinaciji s CSRF-om, u nastavku teksta pobliže su objašnjene ove dvije metode.

2.2.1. XSS

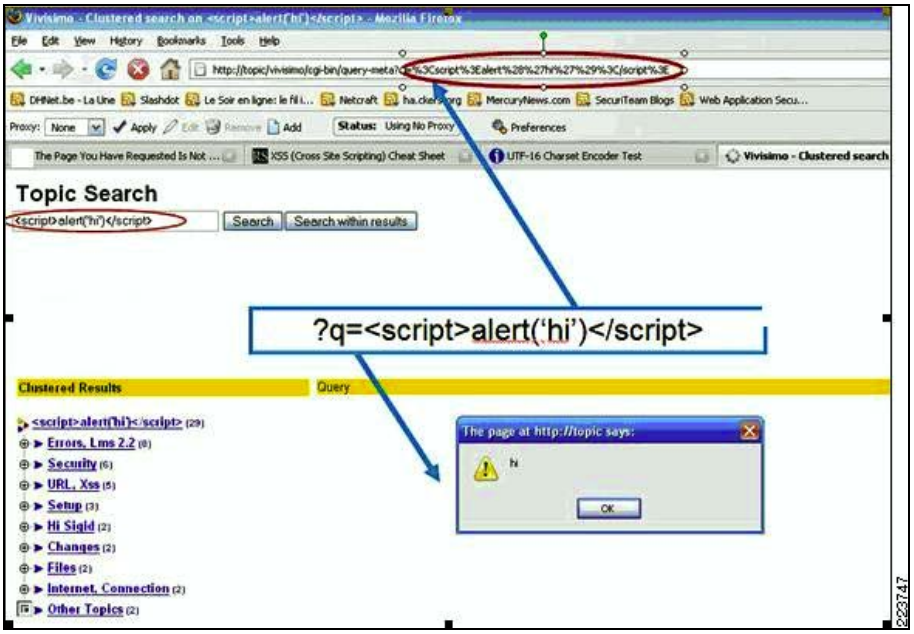
XSS je napad sličan CSRF napadu. Riječ je o metodi koja na ranjivu stranicu podmeće zloćudan sadržaj, a iskorištava nedovoljnu provjeru unosa u programskom kodu stranice. Napadač, primjerice, može podmetnuti posebno oblikovanu poveznicu na ranjivu stranicu. Takva poveznica može sadržavati skriptu koja će izmijeniti sadržaj odredišne stranice (npr. prikazati obrazac za prijavu korisnika). Ukoliko korisnik ranjive stranice iskoristi zloćudnu poveznicu prikazat će mu se podmetnuti sadržaj. Korisnik koji unese podatke u podmetnuti obrazac predaje ih zapravo napadaču (jer je tako napadač definirao u podmetnutoj skripti). Na taj način moguće je otkriti osjetljive podatke. Ovo je tek jedan primjer XSS napada. Primjer jednostavnog XSS koda koji se može umetnuti na ranjivu stranicu jest:

```
<script>alert('poruka');</script>
```

Recimo da je *http://xss.ranjivi.posluzitelj* URL adresa ranjive stranice. Upisivanjem URL zahtjeva koji izgleda slično sljedećem:

http://xss.ranjiva.stranica/zahtjev=<script>alert('poruka');</script>

na stranici će se prikazati prozorčić upozorenja koji može izgledati kao u primjeru na donjoj slici.



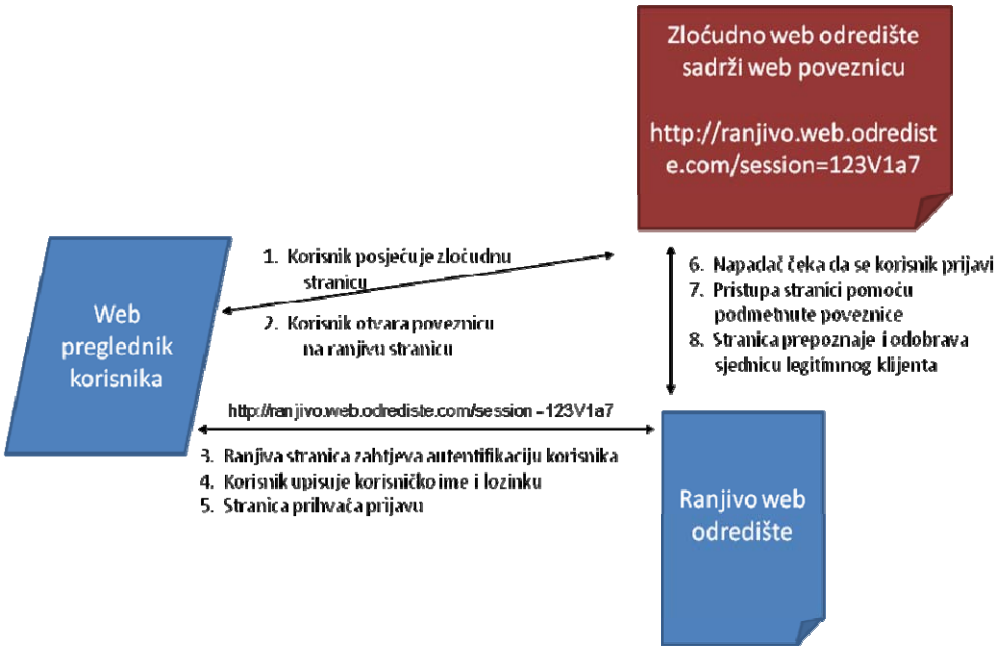
Slika 4. Primjer XSS napada na stranicu
Izvor: Cisco

Načina na koji se ova vrsta ranjivosti može iskoristiti ima mnogo. Prema WASC-u, preko 30% ranjivosti otkrivenih tijekom 2007. bile su XSS ranjivosti. Razlika između XSS-a i CSRF-a je u tome što se kod XSS-a iskorištava povjerenje koje korisnik ima u ranjivu stranicu, odnosno u sadržaj prikazan na njoj. Kod CSRF-a se pak iskorištava povjerenje stranice u korisnika, odnosno u autentičnost podnositelja zahtjeva. Ove dvije ranjivosti mogu se iskoristiti u istom napadu. Primjer napada koji koristi i XSS i CSRF metodu je *Samy* crv. Profil korisnika koji je zaražen XSS crvom prikazivao je poruku *"but most of all, samy is my hero"*. S druge strane, posjet drugog korisnika zaraženom profilu pokrenuo bi CSRF napad te bi se preko njegovog preglednika isti crv učitao u drugu stranicu, onu koja sadrži njegov profil. Na taj se način crv vrlo brzo širio društvenom mrežom. Šteta od ovog napada sama po sebi nije bila ozbiljna, ali je pokazala ozbiljnost i doseg koji ove dvije ranjivosti u kombinaciji mogu imati.

2.2.2. Fiksacija sjednice

Druga ranjivost slična CSRF napadu je i tzv. fiksacija sjednice, odnosno sjedničkog ključa. Pojedine stranice dopuštaju uspostavljanje sjedničkog ključa preko URL zahtjeva. Napad se može izvesti tako da se korisnika navede na posjetu ranjive stranice i pritom se u URL poveznice podmetne

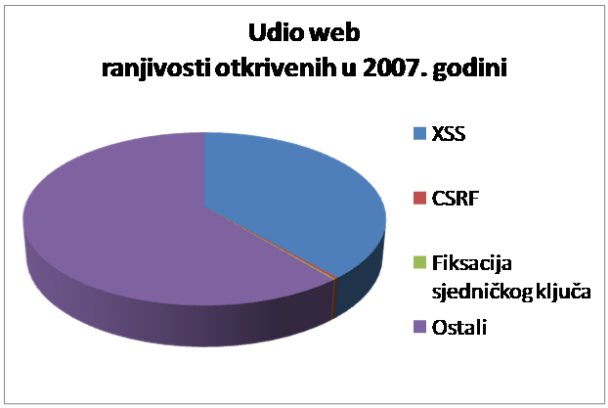
identifikator sjednice. Ako korisnik upotrijebi poveznicu i prijavi se na ranjivu stranicu, podmetnuti sjednički ključ napadač može iskoristiti za pristupanje stranici kao prijavljeni korisnik. Osim toga napad se može izvesti i tako da se uspostavi sjednica sa stranicom i upamti sjednički ključ. Potom se korisnika navede na posjetu te stranice preko uspostavljene sjednice (podmetanjem odgovarajuće poveznice). Nakon što stranica zatraži, a korisnik izvede prijavu, napadač može preko sjedničkog ključa pristupati stranici kao da je prijavljeni korisnik. Još jedan mogući način izvedbe ovog napada je pohranjivanje sjedničkog ključa zloćudnog *web* odredišta u domenu drugog *web* odredišta (u *web* pregledniku koji to dopušta) koje ne mora biti ranjivo. Korisnik, koji potom pristupa ranjivom *web* odredištu, koristi sjednički ključ koji je u *web* preglednik podmetnut prilikom posjeta zloćudnom odredištu. Kao i u prethodno opisanim slučajevima, nakon prijave korisnika, napadač može pristupati ranjivoj stranici pomoću sjedničkog ključa.



Slika 5. Shema napada fiksacije sjednice

Ova vrsta napada može se koristiti u kombinaciji s CSRF-om za oblikovanje zloćudnih URL poveznica preko kojih se stječe neovlašten pristup ranjivom *web* odredištu. Pokazalo se kako nije uvijek potrebno da trenutna sjednica bude zapamćena u *web* pregledniku korisnika, već se mogu koristiti alternativne metode otkrivanja autentikacijskih i sjedničkih atributa. Osim ove metode, moguće je koristiti i napad grubom silom (eng. *brute force*) za otkrivanje sigurnosnih znački (eng. *tokens*). Učinkovitije od toga je otkrivanje podatka vezanih uz povijest pretraživanja koji su sačuvani u *web* pregledniku. Pritom su opasnosti veće ako su takve značke iste tijekom cijele sjednice ili ako je dopušteno korištenje zastarjelih znački, tj. onih predanim u prethodnim *web* obrascima.

Nasuprot XSS napada, ova je ranjivost jednako rijetko iskorištavana kao i sam CSRF što je vidljivo iz statistike WASC-a. Reprezentativna statistika za 2007.g. dana je na slici 6.



Slika 6. Statistika* web *ranjivosti za 2007. godinu
Izvor: WASC

3. CSRF napadi

CSRF napadi mogu se izvoditi na različite načine:

- pomoću zlonamjerno oblikovanih HTML objekata,
- pomoću skriptnog koda umetnutog u HTML (JavaScript, PHP, JScript…) i
- zlouporabom automatskog generiranja zahtjeva u *web* pregledniku (XMLHttpRequest).

Posljednje navedeni način dio je *Web* 2.0 i AJAX tehnologija. Riječ je o naprednim sustavima koji povećavaju raspon mogućnosti i usluga koje pružaju *web* aplikacije. Koliko mogućnosti pružaju programerima koji razvijaju *web* aplikacije, toliko prostora otvaraju i napadačima koji tu istu tehnologiju mogu iskoristiti za izvođenje sofisticiranijih napada. Posebno značenje vezano uz *Web* 2.0 i AJAX ima spomenuta funkcija XMLHttpRequest o kojoj će se detaljnije govoriti u drugom dijelu ovog poglavlja.

3.1. Napadi HTML objektima

CSRF napadi mogu se izvesti na više načina pomoću HTTP GET metode koja služi za dohvat podataka. Klasičan primjer CSRF napada je pomoću HTML objekata. Primjer takvog objekta je IMG koji se koristi za oblikovanje slikovnog *web* sadržaja.

```
<img src="slika.gif" alt="Slika" title="Slika" />
```

U atributu „src“ pohranjen je izvor s kojeg se preuzima slika. Prilikom učitavanja stranice automatski se pristupa tom izvoru kako bi se učitala slika. Podmetanjem zloćudnog koda u taj atribut moguće je poslati zahtjev drugom *web* odredištu. Primjerice, umetanjem sljedećeg koda

```
<img src="http://posluzitelj/stetna_naredba"/>
```

omogućuje se automatsko pokretanje štetne naredbe na poslužitelju. Sve što korisnik treba učiniti jest učitati zloćudnu stranicu u svoj *web* preglednik. Na sličan način mogu se iskoristiti SCRIPT i IFRAME objekti koji također sadrže atribut „src“. U HTML SCRIPT objektu sadržan je kod ili se kodu udaljeno pristupa preko atributa „src“. IFRAME objekti omogućuju učitavanje drugih *web* stranica u poseban okvir unutar trenutne stranice. U oba ova slučaja kao izvor udaljenog koda može biti podmetnut zloćudan zahtjev prema ranjivoj stranici.

```
<iframe src="http://posluzitelj/stetna_naredba"/>
```

ili

```
<script src="http://posluzitelj/stetna_naredba"/>
```

3.2. Napadi skriptnim kodom

Skriptni kod također može sadržavati ranjivosti. Posebice su poznate ranjivosti JavaScript programskog koda. Jednostavan primjer zlouporabe je pomoću objekta *Image()*

```
<script>
  var slika = new Image();
  slika = http://posluzitelj/stetna_nadredba;
<script>
```

ili pomoću objekta ActiveXObject u Microsoftovom JScript skriptnom jeziku

```
<script>
  var xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  xmlhttp.open("POST", 'http://posluzitelj/stetni_kod', true);
  xmlhttp.onreadystatechange = function () {
  if (xmlhttp.readyState == 4)
  {
  alert(xmlhttp.responseText);
  }
  };
  xmlhttp.send(null);
 </script>
```

Skriptni kod ugniježđen u HTML također može biti sredstvo napada. Ukoliko je omogućeno izvođenje skriptnog koda u *web* pregledniku on se pokreće automatski prilikom učitavanja stranice.

3.3. XMLHttpRequest

XMLHttpRequest objekti središnji su objekti AJAX programa. AJAX (eng. *Asynchronous Javascript And XML*) je skupina tehnologija koja omogućuje slanje i obradu HTML zahtjeva u pozadinskom radu programa tako da korisnik ne mora čekati ponovno učitavanje stranice. Klasično slanje HTTP zahtjeva odvija se predavanjem određenog *web* formulara ili otvaranjem *web* poveznice. U pozadinskom radu, HTTP zahtjevi generiraju se automatski u *web* pregledniku bez interakcije s korisnikom. U tu svrhu koriste se XMLHttpRequest objekti.

Primjer uporabe ove funkcije u Internet Exploreru dan je u programskom kodu koji slijedi

```
<script>
  xmlhttp=new XMLHttpRequest() ;
  xmlhttp.open("GET", "http://urlAdresa",true);
  xmlhttp.onreadystatechange = ispisiOdgovor();
  xmlhttp.send(null);

  function ispisiOdgovor(xmlhttp,element_id)
  {
      var element = document.getElementById(element_id);
      if (xmlhttp.readyState == 4)
      {
            var tekst = http.responseText;
            dokument.write.tekst;
      }
  }
</script>
```

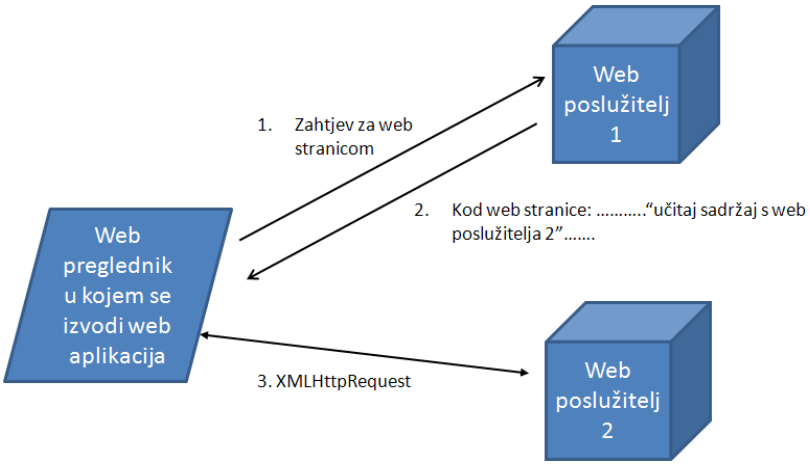
U kodu je prikazano stvaranje novog XMLHttpRequest zahtjeva funkcijom „*open*" i slanje odgovora funkcijom „*send*". Funkcija *ispisiOdgovor* definirana je kao funkcija koja se poziva nakon svake promjene svojstva *onreadystatechange* objekta *xmlhttp*. Radi se o funkciji koja se poziva kada poslužitelj vrati odgovor na zahtjev. Ta funkcija ispisuje odgovor ako je očitana vrijednost 4 svojstva *readyState*, odnosno ako je zahtjev gotov i odgovor spreman za daljnju obradu.

3.3.1. Napredak *Web* 2.0 tehnologije i ranjivosti preglednika

Više puta je spomenuto kako su *web* ranjivosti vezane uz ranjivosti *web* stranica, ali i uz ranjivosti *web* preglednika. Kad su preglednici u pitanju često se radi o podacima koje oni pohranjuju (npr. autentikacijski kolačići, lozinke i korisnička imena). Osim toga *web* preglednici omogućuju automatsko pokretanje određenih zahtjeva kako bi se omogućilo učinkovitije upravljanje *web* sustavima. Tako, primjerice, stranica koja prikazuje kartu određenog područja pokretom miša može pokretati akciju približavanja ili udaljavanja karte. Takva akcija podrazumijeva dohvat *web* sadržaja, a izvodi se izvan korisnikove izravne kontrole. Općenito, u *Web* 2.0 aplikacijama često se koristi automatska komunikacija između većeg broja *web* odredišta kako bi se oblikovale napredne metode rukovanja podacima. Važnu ulogu u tom postupku ima funkcija XMLHttpRequest. Riječ je o funkciji pomoću koje *web* preglednik može pokretati zahtjeve za *web* sadržajem bez interakcije s korisnikom.

3.3.2. XHR i pravilo istog izvora i odredišta

Od prvih inačica funkcija XHR izvedena je tako da poštuje tzv. *„same origin“* pravilo. Riječ je o pravilu koje zahtjeva da se *web* zahtjevi mogu pokretati samo nad stranicama iz kojih su učitani. Tako je *web* odredište s kojeg se učita programski kod jedino odredište prema kojem mogu biti upućeni zahtjevi za sadržajem koji je dio učitanog koda. Na taj način otklanja se mogućnost CSRF napada. Ipak, valja napomenuti kako samo pravilo istog izvora i odredišta ne jamči potpunu sigurnost. Može se zaobići manipulacijama DNS-om ili skrivanjem pravog izvora zahtjeva posebnim oblikovanjem HTML-a. Primjer takvog oblikovanja može biti manipulacija *„Referer“* zaglavljem koje sadrži informaciju o izvoru poruke. Razvojem *Web* 2.0 tehnologija sve je naglašenija potreba za komunikacijom između *web* odredišta bez interakcije korisnika, odnosno za odbacivanjem *„same origin“* pravila. Time se otvaraju naprednije mogućnosti oblikovanja *web* sustava i usluga, ali se i povećava opasnost od CSRF i drugih napada.



Slika 7. Skica rada XHR komunikacije

Inačica standarda	Ograničene metode	Ograničena zaglavlja
30. rujan 2008.	Kao i u prethodnoj inačici	Acces-Control-Request-Headers, Acces-Control-Request-Methods, Authorization, Connection, Cookie, Cookie2, Host, Origin, User-Agent
15. travanj 2008.	CONNECT, TRACK, TRACE	Kao i u prethodnoj inačici
26. listopad 2007.	Kao i u prethodnoj inačici	Kao i u prethodnoj inačici
18. lipanj 2007.	Kao i u prethodnoj inačici	Kao i u prethodnoj inačici te Connection, Content-Transfer-Encoding, Expect, Via
27. veljača 2007.	Kao i u prethodnoj inačici	Kao i u prethodnoj inačici, ali umjesto zanemarivanja javlja se SECURITY_ERR
27. rujan 2006.	Napomena da funkcije trebaju javiti SECURITY_ERR za metode koje nisu implementirane	Zanemaruju se zaglavlja Accept-Charset, Accept-Encoding, Contetn-Length, Expect, Date, Host, Keep-Alive, Referer, TE, Trailer, Transfer-Encoding ili Upgrade
5. travanj 2006.	Nijedna	Nijedno, ali napominje se kako je potrebno ispravno postaviti Host zaglavlje

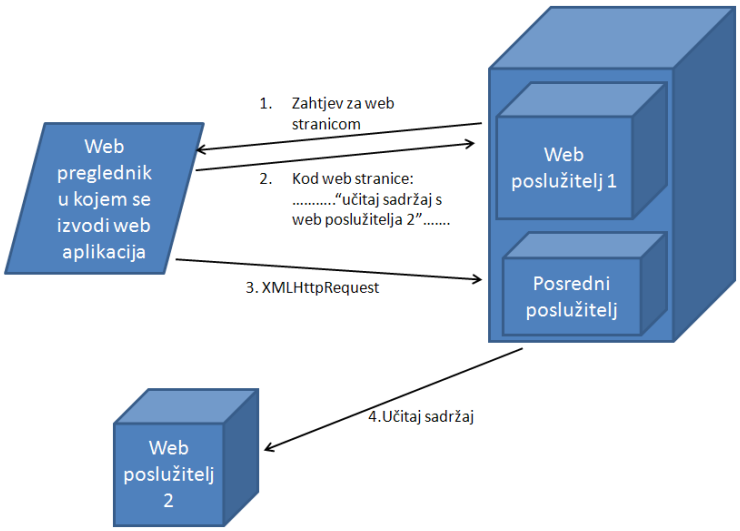
Tablica 1. Sigurnosne izmjene u XHR standardu

Kako bi se otklonila mogućnost zlouporabe uvedene odbacivanjem spomenutog pravila u XRF, ograničava se upotreba pojedinih zaglavlja koja se mogu iskoristiti za CSRF napad. Slična funkcija koju je razvio Microsfot je XDomainRequest. Ona oponaša rad funkcije XHR, ali od početka nije razvijana prema *„same origin"* pravilu, već je briga za sigurnost orijentirana na provjeru i ograničavanje uporabe zaglavlja.

Danas se te funkcije mogu smatrati relativno sigurnima i nemaju istaknutih problema. No, sofisticiranost tehnologije uvijek u stopu prate i napadači tako da su mjere opreza i redovito praćenje sigurnosnih ranjivosti i dalje bitni.

3.3.3. Alternativna rješenja

U *web* preglednicima kod kojih nije podržano slanje XHR zahtjeva drugim *web* poslužiteljima osim onog na kojem je zahtjev generiran, mogu se koristiti alternativna rješenja. Primjerice, može se koristiti posredni (eng. *proxy*) poslužitelj preko kojeg će se zahtjevi slati drugim poslužiteljima.



Slika 8. Komunikacija različitih domena preko posrednog poslužitelja

Osim ovog načina moguće je korištenje digitalnih potpisa na temelju kojih će se generirani kod smatrati sigurnim.

4. Zaštita

Zaštita od CSRF i drugih napada na *web* programe sastoji se od:

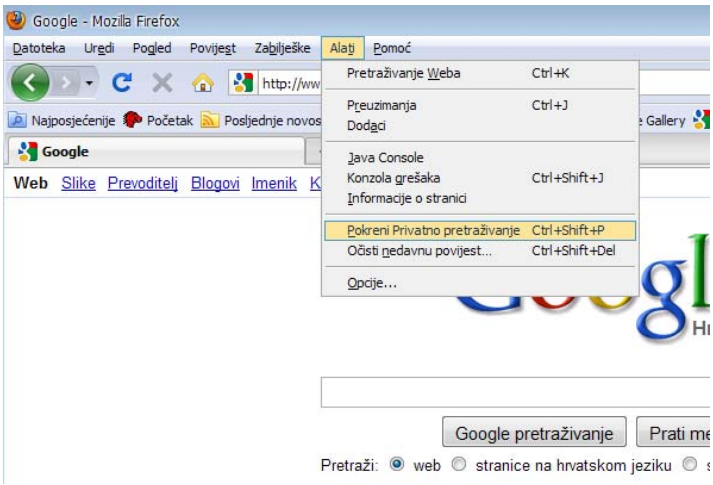
- primjene odgovarajućih principa i mjera zaštite prilikom oblikovanja programa,
- odgovornog rukovanja programom,
- provjere koda i
- ispitivanja rada programa.

OWASP (eng. *Open Web Application Security Project*) je neprofitna organizacija koja se bavi problematikom sigurnosti *web* programa. Objavljuje programe i dokumentaciju vezane uz zaštitu *web* sustava, otkrivanje ranjivosti i sigurnost održavanja. Između ostalih dokumenata sadrži i dokumente sa savjetima za otkrivanje CSRF napada te za zaštitu i prevenciju. Za zainteresirane korisnike, riječ je o specijaliziranom izvoru informacija i alata koji doprinose sigurnosti *web* usluga.

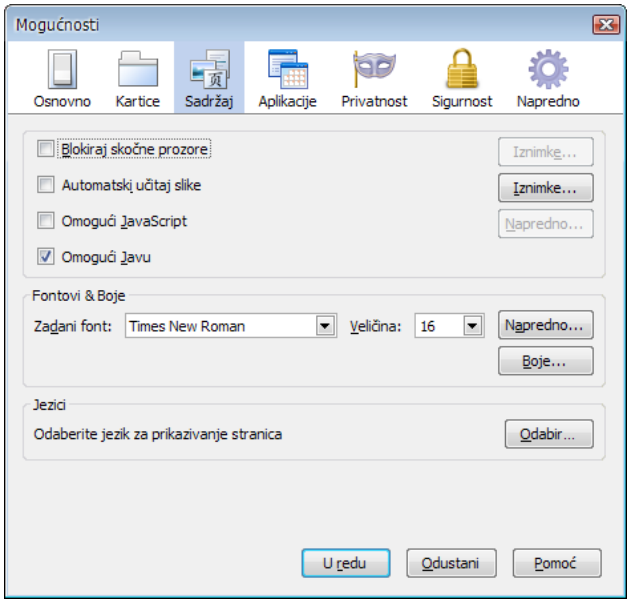
U nastavku poglavlja dan je sažetak savjeta vezanih uz zaštitu s klijentske i poslužiteljske strane te uz način na koji se provjeravaju kod i rad programa.

4.1. Zaštita s korisničke strane

Sama zaštita korisnika u ovom slučaju znači odgovorno korisničko ponašanje odnosno izbjegavanje otvaranja sumnjivih poveznica poput onih u *spam* porukama. Kako bi se spriječilo otkrivanje sigurnosnih znački i autentikacijskih oznaka poželjno je brisanje povijesti pregledavanja ili dodavanje sigurnosnih programskih dodataka u *web* preglednik kao što je SafeHistory za Firefox. Moguće je i korištenje tzv. *„private browsing"* načina rada u kojem se posebno štiti anonimnost korisnika. Primjer mogućnosti koje je moguće isključiti u *web* pregledniku je automatsko pokretanje Javascript koda ili učitavanje slika s vanjskih odredišta. Ove metode nisu omiljene među korisnicima jer smanjuju dostupnu kvalitetu i kvantitetu *web* sadržaja. Na donjim slikama prikazani su izbornici u Mozilla Firefoxu preko kojih se mogu promijeniti neke sigurnosne postavke.



Slika 9. Privatno pretraživanje i web povijest u Firefoxu



Slika 10. Sigurnosne mogućnosti u Firefoxu

4.2. Zaštita poslužitelja web *aplikacije*

Kako bi se smanjila mogućnost CSRF napada sa strane *web* poslužitelja preporučljivo je voditi računa o XSS ranjivostima jer se one mogu koristiti kao podloga za CSRF napade i omogućuju zaobilaženje svih drugih zaštitnih mjera. Kako bi se izbjeglo lažno predstavljanje korisnika, neke od preporučenih mjera zaštite u poslužiteljima *web* programa su:

- ponovno zatražiti prijavu korisnika prilikom svakog kritičnog GET i POST zahtjeva,
- ograničiti vrijeme valjanosti autentikacijskih kolačića,
- provjera izvor poruke (HTTP Referer zaglavlje),
- uvođenje dodatne tajne sigurnosne značke koja se pridružuje identifikacijskim oznakama sjednice i zahtjeva,
- korištenje novih znački u svakoj novoj predaji obrasca čak i unutar iste sjednice,
- odbijanje zastarjelih znački i
- izbjegavanje prikazivanja sjedničkih atributa u URL poveznici. Preporuča se njihovo slanje u skrivenim poljima *web* obrazaca.

Ako je u *web* pregledniku onemogućeno povezivanje različitih domena preko HTTP zahtjeva (pravilo istog izvora i odredišta), mogu se koristiti posredni poslužitelji kako je opisano u poglavlju 3.3.3. Pritom je za zaštitu sigurnosti poželjno u posrednom poslužitelju obavljati provjeru zahtjeva. U suprotnom, ukoliko provjere ne bi bilo, mogući su CSRF napadi i povezivanje proizvoljnih domena. Primjer zaštite može biti tzv. *hardkodiranje* zahtjeva, odnosno upisivanje predviđenih i očekivanih zahtjeva izravno u kod programa tako da nije moguće podmetnuti zahtjeve prema drugim odredištima.

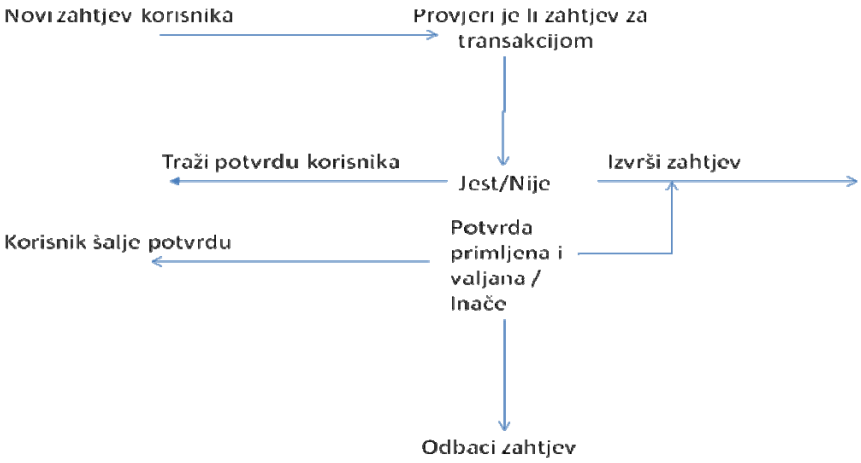
4.3. Ispitivanje ranjivosti

U uvodu poglavlja spomenuto je kako se ranjivosti mogu tražiti izravno u kodu programa ili kroz ispitivanje njegovog rada. U ovom dijelu poglavlja opisani su glavni elementi oba načina na koje je, u slučaju CSRF ranjivosti, važno obratiti pažnju.

4.3.1. Provjera programskog koda

Provjera ranjivosti u kodu najčešće se odnosi na provjeru načina na koji se obrađuju zahtjevi. Pažnju je potrebno obratiti na to je li tretiranje zahtjeva u skladu s njegovim značenjem u poslovnoj domeni. Nisu svi zahtjevi jednako kritični niti jednako osjetljivi. Kod onih koji su osobito visokog rizika, potrebno je nakon zaprimanja svakog novog zahtjeva zatražiti dodatnu potvrdu korisnika. Ona može uključivati i upis lozinke. Sam identifikator korisnika ili sjednice ne smatra se

sigurnom potvrdom jer se lako može lažirati. Ono što je važno jest autentikacija, odnosno provjera identiteta koja se provjerava lozinkama ili složenijim protokolima.



Slika 11. Dijagram obrade zahtjeva

Također kod provjere programskog koda valja obratiti pozornost na to stvaraju li se uz zahtjeve sigurnosne značke, tj. jedinstveni nasumično generirani brojevi koji osiguravaju autentičnost zahtjeva.

4.3.2. Provjera rada stranice

Provjera rada stranice sastoji se od:

- pronalaženja mogućih ranjivih radnji koje se mogu pokrenuti preko URL zahtjeva te
- oponašanja napada.

Pronalaženje mogućih ranjivih točaka izvodi se jednostavno pregledavanjem stranice i isprobavanjem svih interaktivnih mogućnosti. Oponašanje napada može se izvesti na način da se ispitivač sam prijavi na sustav, ukoliko za to ima prava, i preko svog računa ispita CSRF mogućnosti. Ukoliko korisnik nema mogućnosti prijave, potrebno ja izvesti pravi napad na stranicu pomoću kojeg će se dokazati eventualno postojanje CSRF ranjivosti.

Napad se izvodi:

- oblikovanjem zlćudnog URL zahtjeva koji će se podmetnuti korisniku,
- stvaranjem HTML stranice koja će sadržavati zlćudni URL,
- osiguravanjem da je korisnik čiji se račun napada prijavljen na ranjivu stranicu,
- navođenjem korisnika na posjetu ispitne stranice i otvaranje URL poveznice te
- utvrđivanjem je li podmetnuti zahtjev izveden na poslužitelju.

Osim ovog načina, moguće je pratiti rad programa i način na koji se upravlja sjednicom. Ukoliko se upravljanje sjednicom oslanja na korisničke elemente kao što su kolačići i HTTP autentikacija, program je ranjiv na napade. Sigurniji su oni programi koji URL oblikuju uključujući podatke o sjednici koji su nepredvidivi i nedostupni korisniku. Na taj način napadaču se onemogućuje oblikovanje URL zahtjeva koji bi mogao podmetnuti u napadu jer nema pristup svim elementima koji sudjeluju u njegovom stvaranju.

5. Zaključak

Relativno rijetka učestalost CSRF napada može se objasniti složenošću njegova izvođenja. Problem za napadača je što on mora znati točno kada je korisnik prijavljen na ranjivo odredište, zatim što točno želi izvesti na tom odredištu i način na koji će to izvesti. Uz sve to mora navesti korisnika na otvaranje zloćudne poveznice koju mu podmeće. Čak i ako se sve okolnosti poklope, stranice koje mogu pretrpjeti ozbiljnu štetu od ovakvog napada (npr. internetska financijska poslovanja), obično su primjereno zaštićene. Stoga stvarna opasnost od ovakvog napada nije velika, ali zbog njegovog potencijala ne treba je zanemariti. Ukoliko se otvori prostor za CSRF zlouporabe, šteta koju napadač može nanijeti vrlo je ozbiljna.

O sigurnosti i zaštiti od CSRF napada valja voditi brigu odgovornim ponašanjem i sigurnim oblikovanjem programa koji će prikladno provjeravati sve pristigle zahtjeve. Osim toga, valja imati na umu da se Internet neprestano i ubrzano razvija. Nove tehnologije, kao što su Web 2.0 i AJAX proširuju mogućnosti za programere *web* usluga, ali i za napadače. Zato briga za sigurnost ne završava u jednom ciklusu obrazovanja već zahtjeva trajno praćenje razvoja. To se ujedno odnosi i na razvojne programere i na korisnike.

6. Reference

[1] Jason Lam,Johannes B. Ullrich: *Cross Site Request Forgery: What Attackers Don't Want You to Know A Study of Browser Implementations and Security Mechanisms for XMLHttpRequest and XDomainRequest*, http://www.sans.org/reading_room/application_security/protecting_web_apps2.pdf, travanj 2010.

[2] Wikipedia: *Cross-site request forgery*, http://en.wikipedia.org/wiki/Cross-site_request_forgery, travanj 2010.

[3] Wikipedia: Samy(XSS), http://en.wikipedia.org/wiki/Samy_%28XSS%29, travanj 2010.

[4] Wikipedia: *Cross-site scripting*, http://en.wikipedia.org/wiki/Cross-Site_Scripting, travanj 2010.

[5] Wikipedia: *Session fixation*, http://en.wikipedia.org/wiki/Session_fixation, travanj 2010.

[6] Chris Shiflett: *Security Corner*: *Cross-Site Request Forgeries*, http://shiflett.org/articles/cross-site-request-forgeries, travanj 2010.

[7] SecureThoughts.com: *Hacking CSRF Tokens using CSS History Hack*, http://securethoughts.com/2009/07/hacking-csrf-tokens-using-css-history-hack/, travanj 2010.

[8] Robert Auger: *The Cross-Site Request Forgery* (CSRF/XSRF) FAQ, http://www.cgisecurity.com/csrf-faq.html, travanj 2010.

[9] OWASP, http://www.owasp.org/index.php/Main_Page, travanj 2010.

[10] *Web Application Security Statistics* 2008, http://projects.*web*appsec.org/f/WASS-SS-2008.pdf, travanj 2009.

[11] *Web Application Security Statistics* Project 2007, http://projects.*web*appsec.org/f/wasc_wass_2007.pdf, travanj 2009.