Univerzitet Singidunum
Fakultet za informatiku i računarstvo

Identifikacija klijenata u veb aplikacijama

Diplomski rad

Mentor
Dr. Aleksandar Jevremović

Student
Slavoljub Popović (201548/2013)

Beograd, jul 2014.

Sadržaj

1. Uvod 2

2. Osnovi pojmovi, značaj, mogućnosti i implikacije identifikovanja klijenata . 2

2.1 Identifikacija, autentifikacija, autorizacija 3

3. Osobine HTTP protokola 3

3.1 Metode zahteva 4

3.2 Oznake statusa 5

3.3 Perzistentne konekcije 5

3.4 Primer HTTP sesije 6

4. HTTP kolačići 7

4.1 Terminologija i vrste HTTP kolačića 7

4.2 Struktura HTTP kolačića 9

4.3 Lokalno čuvanje podataka u pregledačima 10

4.4 Neovlašćeno preuzimanje kolačića 11

4.5 Ostala tehnička ograničenja kolačića 13

4.6 Alternative za HTTP kolačiće 13

4.7 ETag 15

5. Metodologija kreiranja digitalnog otiska prista (fingerprinting) 16

5.1 Metode bazirane na JavaScript-u 17

5.2 Metode bazirane na dodacima pregledača 18

5.3 Metode bazirane na HTTP zaglavljima i serverske metode 18

5.4 Identifikacija na osnovu liste instaliranih fontova 19

5.5 Identifikacija na osnovu HTML5 osobina 20

6. Implementacija identifikacije klijenata 23

6.1 Algoritam identifikacije 23

6.2 Matematička osnova 24

6.3 Arhitektura rešenja 25

7. Analiza efikasnosti rešenja 30

8. Zaključak 34

9. Literatura 35

1. Uvod

Poznato je da mnogi uređaji poseduju neznatne ali merljive razlike koje omogućavaju da se odredi jedinstveni identifikator uređaja (eng. fingerprint). Kamere, pisaće mašine, kvarcni satovi su među uređajima koji se mogu u potpunosti ili sa određenom pouzdanošću identifikovati samo uz pomoć proizvoda rada uređaja ili komunikacije sa uređajem.

Ideja da se sličan metod primeni u identifikaciji Internet pregledača (browser-a) takođe nije nova. Ona se koristi u nekom obliku od samih početaka Interneta, a posebno od strane kompanija koje prodaju ili distribuiraju oglase, u cilju što bolje analize auditorijuma.

U ovom radu opisaćemo neke od tehnika identifikacije klijenata, implementiraćemo praktično rešenje koje je kombinacija opisanih tehnika i pokazaćemo efikasnost rešenja u realnim uslovima.

2. Osnovi pojmovi, značaj, mogućnosti i implikacije identifikovanja klijenata

Prva detaljna studija o mogućnostima identifikacije pregledača na osnovu podataka koji su dostupni u JavaScript-u i HTTP protokolu urađena je 2010. od strane EFF (Electronic Frontier Foundation). U pitanju je rad „Koliko je jedinstven vaš pregledač?“ [1] Pitera Ekerslija. Radi sprovođenja ove studije napravljen je projekat Panopticlick (https://panopticlick.eff.org/) na kome korisnici mogu da vide koliko je njihov pregledač unikatan. Cilj projekta je da se pokaže da je moguće praćenje korisnika i u slučaju da su opcije privatnosti dobro podešene.

U međuvremenu je došlo do ubrzanog rasta broja korisnika Interneta, ekspanzije veb aplikacija i uopšteno servisa koji se nude na Internetu, kao i šireg prihvatanja novih tehnologija za razvoj veb aplikacija (u prvom redu HTML5 standarda). Zaštita privatnosti je postala je vrlo važno pitanje obzirom da prosečni korisnik već koristi Internet sa više različitih uređaja (desktop računar, „pametni“ telefon, tablet) i količina ličnih podataka koji se prenose preko mreže i čuvaju na različitim servisima je sve veća. Pomenuti podaci osim osnovnih podataka o ličnosti uključuju i podatke o finansijskim transakcijama, kupovini preko Interneta, geolokaciji korisnika i slično.

Važno je napomenuti da je većina tehnika koje se koriste u identifikaciji i praćenju korisnika u potpunosti automatizovana i uglavnom se dešava bez znanja i pristanka korisnika. Ove tehnike su veoma teške i za otkrivanje obzirom da ne ostavljaju vidljive tragove na uređaju korisnika. Dve osnovne oblasti primene identifikacije klijenata su:

- analiza auditorijuma za oglašavanje i
- sprečavanje prevara u finansijskim transakcijama

Da bismo praktično objasnili implikacije navedenog eksperimenta uzmimo za primer neki od popularnih servisa na Internetu npr. Google pretraživanje (primer je potpuno hipotetički). Ukoliko je korisnik prijavljen sa svojim Google nalogom rezultati pretraživanja će biti personalizovani, biće zabeležena istorija upita i slično. Međutim, ako se korisnik odjavi, Google može nastaviti da prati njegove aktivnosti tehnikom koju smo

prethodno opisali tako što je jedinstveni identifikator klijenta povezan sa korisničkim nalogom.

2.1 Identifikacija, autentifikacija, autorizacija

Potrebno je razgraničiti termine koji su često u upotrebi i imaju određene sličnosti u značenju:

- *Identifikacija klijenta* predstavlja, u opštem slučaju, dodeljivanje identifikatora (ne nužno jedinstvenog) pregledaču. Na ovaj način ne dolazi do stvarne identifikacije korisnika u smislu otkrivanja identiteta odnosno ličnih podataka, već pre možemo govoriti o nekoj vrsti digitalnog otiska prsta. Ukoliko se ne koristi u kombinaciji sa autentifikacijom ili nekom drugom vrstom povezivanja sa identitetom korisnika identifikacija klijenta se može smatrati potpuno bezbednom sa stanovišta privatnosti.
- *Autentifikacija* predstavlja proces predstavljanja (identifikovanja) korisnika informacionom sistemu. Autentifikacija se obično vrši pomoću određenih informacija poznatih samo korisniku i sistemu. U kontekstu veb aplikacija autentifikacija se obično obavlja HTTPS protokolom i kao rezultat obično se snimaju kolačići na strani klijenta i na osnovu njih se vrši identifikacija sesije na strani servera. Kao što je već pomenuto veb aplikacija može da izvrši povezivanje identifikatora klijenta i identiteta korisnika, tako da suštinski postane nebitno da li se korisnik odjavio iz sistema.
- *Autorizacija* je provera prava pristupa korisnika ili grupe korisnika određenim resursima unutar informacionog sistema. Autorizacija nije od značaja za identifikaciju klijenata obzirom da se vrši samo u okviru aktivne sesije.

U nastavku ovog rada pominjaćemo identifikaciju korisnika i/ili klijenta u različitim kontekstima. Osim ako drugačije ne naglasimo pod identifikacijom korisnika podrazumevaćemo dodeljivanje identifikatora pregledaču koji korisnik koristi. Ovakav pristup smatraćemo „fer“ upotrebom indentifikovanja obzirom da je u pogledu realnog identiteta korisnika potpuno bezbedan (tj. u potpunosti poštuje privatnost korisnika).

3. Osobine HTTP protokola

U strukturi Internet protokola HTTP (HyperText Transfer Protocol) nalazi se u aplikacionom sloju i služi kao osnova World Wide Web-a. HTTP se oslanja na TCP (Transfer Control Protocol). HTTP funkcioniše na principu zahtev-odgovor (klijent šalje zahtev, server šalje odgovor).

Pregledač veba (web browser) obično igra ulogu klijenta, a aplikacija koja se nalazi na računaru koji hostuje određeni veb sajt predstavlja server. Server omogućava pristup resursima (HTML fajlovi, slike i drugi sadržaji), izvršava druge funkcije zahtevane od strane klijenta i šalje klijentu odgovor (response message). Odgovor servera sadrži informaciju o statusu zahteva u zaglavlju odgovora (response header) i traženi sadržaj u telu odgovora (response body).

Pregledač veba predstavlja primer tzv. korisničkog agenta (user agent, UA). Korisnički agenti uključuju i programe za indeksiranje sadržaja (web crawlers), glasovne pregledače, mobilne aplikacije i druge programe koji pristupaju, koriste i prikazuju sadržaje na vebu.

HTTP je dizajniran tako da dozvoljava posrednicima u mrežnoj komunikaciji da poboljšavaju komunikaciju klijenta i servera. Sajtovi koji imaju veliki broj posetilaca često koriste servere za keširanje koji isporučuju sadržaj umesto glavnih servera i na taj način poboljšavaju vreme odziva. Pregledači veba takođe keširaju resurse kojima su pristupali i koriste lokalne verzije kad god je to moguće da bi se smanjio mrežni saobraćaj. Tzv. HTTP proksi serveri (proxy servers) preko kojih privatne mreže pristupaju spoljnim sadržajima služe kao posrednici u komunikaciji klijenata bez javnih adresa sa javnim veb serverima.

HTTP resursi se identifikuju i lociraju na mreži uz pomoć uniformnih identifikatora resursa (Uniform Resource Identifiers – URIs) ili uniformnih lokatora resursa (Uniform Resource Locators – URLs), koristeći http ili https URI šeme. URI i hiperlinkovi u HTML dokumentima formiraju mrežu međusobno povezanih hipertekstualnih dokumenata.

Trenutno je u upotrebi HTTP/1.1 revizija HTTP protokola. U odnosu na originalnu HTTP/1.0 specifikaciju u kojoj je bila predviđena zasebna konekcija za svaki zahtev za resursima, u HTTP/1.1 jedna konekcija se može iskoristiti više puta za preuzimanje slika, skriptova i drugih sadržaja nakon što je HTML dokument učitan. Na taj način je značajno unapređena brzina prenosa.

HTTP sesija predstavlja sekvencu zahtev-odgovor transakcija između klijenta i servera. Klijent inicira zahtev uspostavljajući TCP konekciju na određeni port na serveru (obično port 80). Server „sluša“ na tom portu i očekuje zahteve od klijenata. Nakon što je zahtev primljen server vrši obradu zahteva i šalje klijentu status (npr. „`HTTP/1.1 200 OK`“) i poruku koja predstavlja odgovor na zahtev. Telo odgovora je obično sam resurs koji je zahtevan, mada može biti i poruka o grešci ili neka druga informacija. HTTP je protokol koji ne pamti stanje (stateless protocol), što znači da sam protokol odnosno server nije dizajniran tako da čuva informacije ili statuse pojedinačnih korisnika tokom trajanja više uzastopnih zahteva. Zbog toga se u veb aplikacijama obično implementiraju sesije na strani servera koristeći HTTP kolačiće ili skrivene promenljive unutar veb formi.

3.1 Metode zahteva

HTTP definiše metode pomoću kojih se može naznačiti kakvu akciju treba uraditi sa traženim resursom. Šta je tačno resurs, odnosno da li se radi o već postojećim podacima ili podacima koji su dinamički kreirani, zavisi od implementacije servera. U originalnoj HTTP specifikaciji bili su definisani GET, POST i HEAD metodi, u 1.1 specifikaciji dodato je još 5 metoda: OPTIONS, PUT, DELETE, TRACE i CONNECT (RFC2616). Klijenti mogu da koriste bilo koji metod, a serveri mogu biti konfigurisani da podržavaju bilo koju kombinaciju ovih metoda. Ukoliko je metod nepoznat serveru, on će biti tretiran kao nesiguran metod uz odgovarajuću poruku o grešci. Takođe, nema ograničenja u broju novih metoda koji mogu biti definisani tako da ova specifikacija omogućava nadogradnje bez ugrožavanja postojećih funkcionalnosti.

- GET metodom zahteva se prezentacija treženog resursa. GET zahtevi trebalo bi da služe samo za preuzimanje podataka i ne bi trebalo da imaju druge efekte
- HEAD metodom zahteva se odgovor identičan odgovoru na GET zahtev ali bez tela poruke. Ovo je naročito korisno ukoliko želimo da preuzmemo meta-podatke o određenom resursu koji se nalaze u zaglavlju odgovora, ali ne želimo da preuzimamo sam resurs
- POST metodom zahteva se da server prihvati prateći sadržaj i sačuva ga pod priloženi URI. Ako URI označava već postojeći resurs onda se vrši ažuriranje tog resursa, inače se vrši kreiranje novog resursa

- DELETE metodom zahteva se brisanje određenog resursa
- OPTIONS metodom dobija se lista HTTP metoda koje server podržava za određeni URL. Ova metoda se može koristiti za proveru funkcionalnosti servera prosleđivanjem „*“ umesto određenog resursa
- CONNECT metodom se konvertuje konekcija zahteva u transparentni TCP/IP tunel, obično da bi se olakšala SSL-kriptovana komunikacija kroz HTTP proksi koji nije kriptovan

HTTP serveri bi trebali da implemetiraju makar GET i HEAD metode i, kad god je to moguće, OPTIONS metod.

Neki od metoda (npr. HEAD, GET, OPTIONS) definisani su kao sigurni metodi, što znači da su namenjeni samo za preuzimanje informacija i da pomoću njih ne bi trebalo da se promeni stanje servera. Drugim rečima ove metode ne bi trebalo da imaju bočne efekte osim relativno bezopasnih kao što su logovanje i keširanje. Nasuprot tome metodi POST, PUT, DELETE su namenjeni za izmene podataka i mogu imati različite bočne efekte (npr. finansijske transakcije ili slanje emaila). Ovo je značajno zbog programa za indeksiranje sadržaja ili drugih aplikacija koje vrše automatizovano pristupanje resursima. Programi za indeksiranje koji se pridržavaju standarda koriste samo sigurne metode zahteva u svom radu. Međutim, dešava se da zbog nepridržavanja preporuka koje smo opisali aplikacije koriste sigurne metode za izmenu stanja na serveru što može dovesti do grešaka koje su vrlo teške za otkrivanje, ali i do direktne štete nastale „slučajnim“ pristupom resursima.

3.2 Oznake statusa

Prema HTTP specifikaciji prva linija HTTP odgovora naziva se statusna linija i sadrži numeričku oznaku statusa (npr. 404) i tekstualni opis statusa (npr. „Not Found“). Način na koji korisnički agent vrši obradu odgovora zavisi primarno od oznake statusa, dok je ostatak zaglavlja odgovora od manjeg značaja. Oznake statusa mogu biti proizvoljne, i ako korisnički agent ne prepoznaje status on može koristiti prvu cifru oznake da prepozna generalni tip odgovora.

Standardni tekstualni opis statusa je samo preporuka i može biti zamenjen proizvoljnom frazom. Tekstualni opis statusa služi da bi korisniči agent mogao da pruži dodatne informacije korisniku u slučaju da status označava grešku. Oznake statusa podeljene su u 5 grupa:

- informativne 1XX
- uspešne 2XX
- redirekcija 3XX
- greška na klijentu 4XX
- greška na serveru 5XX

3.3 Perzistentne konekcije

Kao što je prethodno pomenuto trenutni HTTP standard omogućava ponovno korišćenje već ostvarene konekcije. Ovo je moguće uz pomoć tzv. keep-alive mehanizma. Takve konekcije nazivaju se perzistentne i njihovim korišćenjem izbegnuto je da se ponovo radi „trostruko rukovanje“ prilikom uspostavljanja TCP konekcije. Na taj način

brzina zahteva (i odgovora) je značajno povećana. Drugi pozitivni efekat perzistentnih konekcija je da sa prolaskom vremena TCP konekcija postaje sve brža zbog tzv. slow-start mehanizma.

HTTP/1.1 definiše nekoliko značajnih tehnika kojima se dobija na uštedi protoka i brzini prenosa. To su paketno enkodiranje prenosa (chunked transfer encoding) uz pomoć koga je moguće da se podaci prenose u jednom kontinuiranom toku (streaming) i paralelizacija zahteva (HTTP pipelining) koji omogućava klijentima da pošalju više zahteva pre nego što dobiju sve odgovore. Još jedno poboljšanje protokola je i serviranje bajtova koje omogućava da server pošalje deo resursa koji je eksplicitno zahtevan od strane klijenta.

3.4 Primer HTTP sesije

Prikazaćemo kako izgleda primer razmene poruka klijenta i servera HTTP protokolom u uzvornom obliku.

Zahtev klijenta

```
POST /index.html HTTP/1.1
Host: www.example.com
```

Odgovor servera

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
ETag: "3f80f-1b6-3e1cb03b"
Content-Type: text/html; charset=UTF-8
Content-Length: 131
Accept-Ranges: bytes
Connection: close

<html>
<head>
  <title>An Example Page</title>
</head>
<body>
  Hello World, this is a very simple HTML document.
</body>
</html>
```

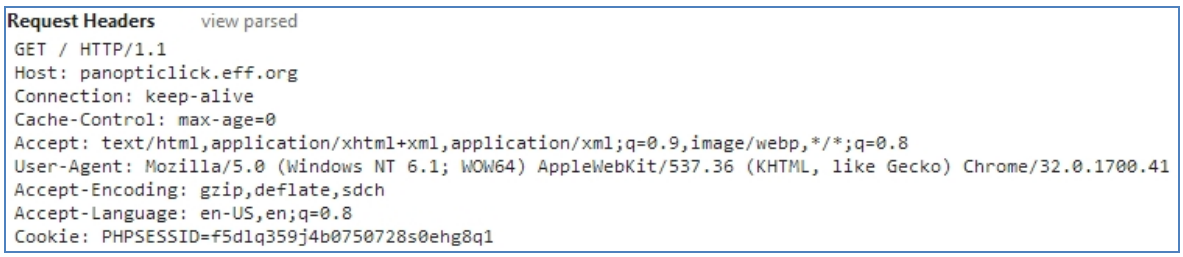
Zahtev klijenta sadrži naziv HTTP metode (kao što je objašnjeno u tački 3.1), adresu zahtevanog resursa, verziju protokola i naziv servera (sajta) od koga se zahteva resurs. Odgovor servera sadrži zaglavlje i telo poruke odvojene sa dva blanko reda. Telo poruke je u ovom slučaju HTML dokument. Zaglavlje odgovora sadrži razne podatke od kojih je za nas najznačajniji ETag (entity tag) koji služi za poređenje trenutne i keširane verzije resursa. ETag se može smatrati jedinstvenim identifikatorom (checksum) resursa, a kao što ćemo videti može se iskoristiti i u svrhe praćenja korisnika. Ostali podaci iz

zaglavlja daju razne instrukcije klijentu kako da obradi sadržaj (`Content-Type`, `Content-Length`), koje su mogućnosti servera (`Accept-Ranges`), kao i koje akcije će biti preduzete nakon slanja sadržaja (`Connection`).

Većina linija iz zaglavlja su opcione, a odgovarajući parametri imaju podrazumevane vrednosti ili se određuju dinamički. Npr. ako nedostaje `Content-Length` on će se izračunati na strani klijenta, a prijem podataka se vrši dok se konekcija ne prekine.

4. HTTP kolačići

Kolačići, HTTP kolačići ili veb kolačići predstavljaju malu količinu informacija (u tekstualnom obliku) koju veb server (odnosno veb sajt) može da pošalje pregledaču, zajedno sa instrukcijama na koji način će se snimiti i koristiti prilikom budućih pregledanja sajta. Obično se kolačići šalju zajedno sa svakim HTTP zahtevom nakon što su prvi put prihvaćeni i snimljeni (slika 1). Obzirom da HTTP protokol ne podržava prenos stanja kolačići se koriste kao mehanizam pomoću koga se to postiže. Primeri u kojima je potreban prenos stanja sreću se u svakoj veb aplikaciji koja ima potrebu za identifikovanjem korisnika ili pruža funkcionalnost koja se ne može obaviti jednim HTTP zahtevom (npr. korpa za kupovinu i sl.).



Slika 1 Primer HTTP zaglavlja sa postavljanjem kolačića

Iako kolačići ne mogu da prenose viruse ili da instaliraju maliciozne programe na klijentima, oni se mogu koristiti za praćenje i snimanje istorije pregledanja sajtova i drugih aktivnosti korisnika što predstavlja ugrožavanje privatnosti korisnika. Zbog toga je od 2011. zakonski propisano u Evropskoj uniji i Sjedinjenim Američkim Državama, da sajt mora da obavesti korisnika i zatraži eksplicitnu dozvolu za korišćenje odnosno snimanje kolačića na korisnikov računar. Kolačići takođe mogu da čuvaju lozinke i brojeve kreditnih kartica koje korisnici unose kroz forme u veb pregleadačima, što takođe predstavlja veliki sigurnosni rizik.

Primarna primena kolačića je za autentifikaciju korisnika, što predstavlja esencijalnu funkciju modernog veba. Uz pomoć kolačića za autentifikaciju sajtovi mogu da odrede da li je korisnik prijavljen ili ne i ako jeste koji je njegov korisnički nalog. Bez ovog mehanizma sajtovi nemaju mogućnosti da znaju kako i kome da pošalju određene osetljive informacije, odnosno da traže od korisnika da se autentifikuju. Sigurnost kolačića za autentifikaciju uopšteno zavisi od sigurnosti sajta koji ih šalje kao i od sigurnosti korisnikovog veb pregledača.

4.1 Terminologija i vrste HTTP kolačića

Obzirom da se identifikovanje klijenata i generisanje digitalnog otiska prsta dobrim delom oslanja na osobine kolačića, izložićemo detaljnije terminologiju, odnosno vrste HTTP kolačića:

- *Sesijski kolačići,* koji se takođe nazivaju i privremeni čuvaju se u radnoj memoriji i to samo za vreme trajanja sesije korisnika na određenom sajtu. Oni se kreiraju ukoliko nije postavljen datum isticanja ili interval isticanja prilikom kreiranja kolačića. Pregledači brišu ove kolačiće u momentu zatvaranja prozora pregledača
- *Perzistentni kolačići* ostaju sačuvani i nakon zatvaranja pregledača, odnosno prestanka sesije korisnika. Ako perzistentni kolačić ima postavljen `Max-Age` atribut npr. na 1 godinu onda u roku od jedne godine od momenta kreiranja kolačić će biti poslat na sajt sa koga je postavljen prilikom svake posete korisnika tom sajtu. Ovo se može koristiti da se zabeleže razni parametri koji su od značaja za praćenje korisnika pa se ovi kolačići nazivaju i kolačići za praćenje (tracking cookies)
- *Sigurni kolačići* imaju postavljen sigurnosni atribut `secure` i koriste se samo preko sigurne konekcije (HTTPS). Na ovaj način kolačić je uvek kriptovan prilikom slanja na server, tako da je smanjena mogućnost čovek-u-sredini napada. Pored toga svi kolačići podležu pravilu istog izvora (same-origin) koje podržavaju svi moderni pregledači a koje zabranjuje slanje kolačića na bilo koji drugi sajt osim onog sa koga su inicijalno snimljeni
- *Kolačići ograničeni na HTTP* - `HttpOnly` atribut je podržan od većine modernih pregledača i ako je postavljen označava da je pristup kolačiću ograničen samo na slanje putem HTTP(S) protokola. To praktično znači da se takvim kolačićima ne može pristupiti iz JavaScript-a. Ovo ograničenje sprečava, ali ne elimniše u potpunosti rizik od preuzimanja kolašića tzv. cross-site scripting tehnikom
- *Kolačići od "traće strane"* – Originalni kolačići potiču sa domena (ili poddomena) koji je prikazan u polju za adresu pregledača (slika 2). Kolačići od „treće strane" bivaju postavljeni od strane sajtova čiji sadržaj koristi sajt prikazan u polju za adresu pregledača (se na ovaj način prikazuju reklame, video sadržaj, alati za deljenje na društvenim mrežama i sl.). U većini modernih pregledača postoji opcija za isključivanje kolačića od „treće strane". Radi slikovitijeg prikaza uzmimo da domen show.ads.com služi za prikazivanje reklama i da sajtovi www.site1.com i www.site2.com prikazuju reklame sa show.ads.com



Slika 2 Kolačići postavljeni od treće strane

Postavljanjem kolačića show.ads.com automatski dobija informaciju o korisnicima koji su posetili jedan ili oba navedena sajta i to apsolutno bez znanja i pristanka korisnika

- *Super kolačić* je kolačić kome je izvor domen najvišeg nivoa (kao što je .com) ili državni domen (kao što je .co.uk). Pregledači blokiraju ovakve kolačiće zbog sigurnosnih problema koji nastaju njihovim korišćenjem. U slučaju da su ovi kolačići omogućeni napadač bi mogao da uz pomoć malicioznog sajta preuzme identitet korisnika drugih sajtova na istom domenu najvišeg nivoa (u slučaju .com domena to znači da bi svaki sajt koji se završava sa .com mogao biti ugrožen). Što se tiče državnih domena najvišeg nivoa stariji pregledači mogu biti ranjivi na ovakve napade obzirom da ne sadrže ažurne liste domena najvišeg nivoa
- *Super kolačići (drugo značenje)* – termin super kolačići (supercookies) se koristi i za tehnike praćenja koje se ne zasnivaju na HTTP kolačićima. Primer za to su sinhronizovanje kolačića koji rekreiraju MUID (Machine Unique IDentifier) kolačiće (oni su pronađeni na Majkrosoftovim sajtovima, ali je Majkrosoft ubrzo popravio ovaj propust) kao i Etag kolačići
- *„Zombi" kolačići (zombie cookie i evercookie)* – neki kolačići se automatski rekreiraju nakon što ih korisnik obriše i oni se nazivaju „zombi" kolačići. Ovo se postiže tako što se kolačiči snimaju na nestandardne lokacije uz pomoć Flash dodatka, HTML5 lokalnog skladišta kao i drugih (ne tako široko dostupnih mehanizama). Kada se detektuje da kolačić nije prisutan poseban skript se koristi da uradi rekreiranje tražeći kolačić na rezervnim lokacijama

4.2 Struktura HTTP kolačića

Prema specifikaciji RFC6265 [2] minimalna veličina kolačića koju pregledači treba da podržavaju je 4KB. Prema pomenutoj specifikaciji HTTP kolačić poseduje 7 atributa:

1. ime – identifikator kolačića, daje mogućnost da u jednom HTTP zaglavlju ima više kolačića
2. vrednost – vrednost kolačića
3. datum važnosti – prema datumu važnosti kolačići se dele na sesijske i perzistentne. Sesijski kolačići brišu se na kraju sesije (zatvaranje pregledača), perzistentni traju do datuma važnosti. Važno je napomenuti da ukoliko je dostignuto globalno ograničenje broja kolačića ili ograničenje za zadati domen onda dolazi do automatskog brisanja kolačića (obično se primenjuje princip da se brišu kolačići koji najduže nisu korišćeni)
4. domen – ime domena koji se mora pojaviti u HTTP zahtevu da bi se poslao kolačić. Prilikom provere da li domen kolačića odgovara trenutnom domenu pregledač radi proveru s desna na levo. Ovo znači da ukoliko je domen u kolačiću npr. „google.com“ kolačić će biti poslan i na „mail.google.com“ i na „maps.google.com“ i sl.
5. putanja – putanja na domenu koja mora biti prisutna u HTTP zahtevu da bi kolačić bio poslat. Prilikom provere putanje radi se provera s leva na desno. Npr. ukoliko je u kolačiću postavljena putanja „/blog” onda će cookie biti poslat i za sve putanje koje počinju sa „/blog“ npr. „/blogroll“, „/blog/some-title“ i sl.
6. sigurna konekcija – ukoliko je prosleđen atribut `secure` kolačić će biti poslat samo preko HTTPS protokola. U praksi, međutim, nikada ne treba koristiti kolačiće za prenos poverljivih ili sigurnosno osetljivih informacija zato što su kolačići, bez obzira da li je postavljen atribut `secure` ili ne podložni raznim vrstama napada
7. samo HTTP – ukoliko je prosleđen atribut `HttpOnly` onda se kolačiću ne može pristupiti iz JavaScript-a (preko `document.cookie`)

Koristeći EBNF notaciju predstavićemo postavljanje i prenos HTTP kolačića [3]:

```
Set-Cookie: name=value[; expires=date][; domain=domain][; path=path][;
secure][; HttpOnly]
```

```
Cookie: name1=value1[; name2=value2][; name1=value1]
```

4.3 Lokalno čuvanje podataka u pregledačima

HTTP kolačići obično se čuvaju kao tekstualni fajlovi na specijalno određenoj lokaciji od strane pregledača (u nekim pregledačima korisnik može da odredi ovu lokaciju). Savremeni pregledači (sa podrškom za HTML5) imaju standardizovano skladištenje i drugih podataka na klijentskom uređaju.

Iako pregledači imaju ograničen pristup lokalnom hard disku i ostalim hardverskim resursima računara postoje tehnike kojima se HTTP kolačići snimaju na različite lokacije sa ciljem da se onemogući njihovo brisanje i na taj način poveća preciznost u praćenju korisnika. Neki autori ovakve kolačiće nazivaju i superkolačići (supercookies) Navešćemo neke od ovih tehnika [4]:

- Local Shared Objects (Flash Cookies)
- Silverlight Isolated Storage

- Čuvanje kolačića kodiranih u RGB vrednosti autogenerisanih, keširanih PNG slika, HTML5 kanvas tag se koristi da se uradi dekodiranje
- Čuvanje kolačića u istoriji pregledanih stranica
- window.name keširanje
- Internet Explorer userData storage
- HTML5 Session Storage
- HTML5 Local Storage
- HTML5 Global Storage
- HTML5 Database Storage via SQLite

Ovo praktično znači da u nekim slučajevima možemo da sačuvamo podatke na više od 10 različitih lokacija izuzimajući standardno čuvanje HTTP kolačića. Ako uzmemo u obzir da prosečan korisnik pregledača koristi podrazumevana podešavanja, možemo zaključiti da postoje značajne, mada malo poznate, pretnje za privatnost korisnika.

Iako tehnike čuvanja HTTP kolačića na više lokacija možemo smatrati malicioznim u savremenim pregledačima ne postoje zabrane (nekad su čak i nemoguće) ovih tehnika. To znači da nedobronameran servis na Internetu može koristiti prethodne tehnike i za efikasniju identifikaciju klijenata.

4.4 Neovlašćeno preuzimanje kolačića

Kao što smo već pomenuli u uvodnom razmatranju ove tačke većina sajtova koristi kolačiće kao jedine identifikatore korisničkih sesija i to najviše zato što ostale metode identifikacije korisnika imaju ograničenja ili su manje sigurne. Krađom kolačića iz pregledača korisnika može se preuzeti identitet korisnika a samim tim i neautorizovano pristupati određenim resursima. Upravo je tema kojom se ovaj rad bavi jedan od mogućih načina kojim se povećava sigurnost korisničkih sesija u veb pregledačima zato što pored identifikatora sesije možemo koristiti i identifikator klijenta (pregledača) koji ne mora da se čuva na strani klijenta u obliku kolačića (ili bilo kom drugom obliku).

Ukratko ćemo prikazati neke od scenarija preuzimanja kolačića i korisničke sesije koji se mogu upotrebiti na sajtovima koji se oslanjaju samo na kolačiće za identifikaciju korisnika:

Prisluškivanje mrežnog saobraćaja

Ukoliko postoji učesnik mrežnog saobraćaja koji može da presretne komunikaciju korisnika i servera (npr. nekriptovana otvorena Wi-Fi konekcija) onda može doći i do preuzimanja kolačića. Ukoliko se komunikacija korisnika i servera vrši standardnim HTTP protokolom kolačići se šalju kao običan tekst tako da napadač ima lak zadatak. Nakon preuzimanja kolačića napadač može da se lažno predstavi kao pravi korisnik i izvede razne maliciozne akcije kao što je transfer novca sa bankovnog računa žrtve i slično. Da bi se izbegla ova vrsta napada treba uvek koristiti HTTPS protokol i postaviti `secure` atribut kolačića tako da se kolačić šalje samo kriptovanim kanalom.

Lažiranje DNS-a

Lažiranjem DNS-a (Domain Name System) napadač prouzrokuje da DNS server kešira lažni DNS unos, npr. f12345.www.example.com koji pokazuje na IP adresu kontrolisanu od strane napadača. Zatim napradač šalje potencijalnoj žrtvi URL do slike,

npr. http://f12345.www.example.com/img_4_cookie.jpg. Otvaranjem ove slike žrtva šalje sve kolačiće sa domena www.example.com na server napadača.

Ovaj problem bi trebalo da bude i nadležnosti Internet provajdera zato što treba povećati sigurnost DNS servera, ali i www.example.com treba da koristi samo sigurne kolačiće. Korišćenjem sigurnih kolačića pregledač žrtve neće poslati kolačiće na nesiguran domen. Napadač se može odlučiti da koristi HTTPS ali bez validnog SSL sertifikata napad neće biti efikasan zato što će pregledač žrtve prikazati upozorenje da je sajt potencijalno nesiguran.

Cross-site scripting

JavaScript koji je prisutan u svim modernim pregledačima, kao skripting jezik i platforma za razvoj aplikacija u pregledačima ima određene mehanizme za čitanje kolačića i slanje na proizvoljne sajtove. Ovo se vrlo često koristi na sajtovima koji omogućavaju korisnicima da postuju HTML koji drugi korisnici mogu da vide (forumi, komentari i sl.)

Kao primer navešćemo link koji napadač može da pošalje na www.example.com:

```
<a href="#" onclick="window.location=
'http://attacker.com/stole.cgi?text='+escape(document.cookie); return
false;">Click here!</a>
```

To je upravo primer cross-site scripting-a: kolačić koji bi trebalo da se razmenjuje samo između klijenta i servera koji ga je inicijalno kreirao prosleđuje se nepoznatom serveru. Napad je tehnički veoma jednostavan jer se zasniva na prostoj činjenici da JavaScript može da pročita kolačiće, a ostatak napada se svodi na iskorišćavanje nepažnje i radoznalosti korisnika. U slučaju da se link postuje npr. na https://www.example.com/ onda će i sigurni kolačići biti poslati i to kao običan tekst.

Cross-site scripting je konstantna pretnja jer uvek ima malicioznih korisnika koji pokušavaju da pronađu propuste u aplikacijama, i zato u sigurnosnim proverama prilikom testiranja aplikacija treba posebno obratiti pažnju na ovu vrstu napada. Pored toga najbolja odbrana protiv ove vrste napada je da se postavi HttpOnly atribut i na taj način onemoguće skriptovi na strani klijenta da pristupe kolačićima.

Cross-site scripting – proksi zahtev

Ova vrsta napada je karakteristična za starije pregledače. U pitanju je iskorišćavanje sigurnosnog propusta koji omogućava napadaču da preusmeri zahtev koji je upućen pomoću `XMLHttpRequest` (tzv. Ajax zahtev). Npr. žrtva čita napadačev post na www.example.com i napadačev skript se izvršava u pregledaču žrtve. Skript generiše zahtev na www.example.com sa proksi serverom attacker.com. Pošto je zahtev regularan kolačići će biti poslati ali pošto zahtev ide preko servera koji kontroliše napadač on će preuzeti kolačiće od žrtve. Korišćenje sigurnih kolačića sprečava ovaj napad.

Cross-site request forgery

Pretpostavimo da je Bob korisnik koji pregleda jedan od Internet foruma, gde je drugi korisnik, Alisa, ostavila poruku. Pretpostavimo da je Alisa u poruci kodirala HTML

sa referencom na sliku koja u atributu koji pokazuje putanju do slike ima specijalni link do sajta Bobove banke:

```
<img src="http://bank.example.com/withdraw?acct=bob&amount=10000&for=alice">
```

Ako je Bob već bio ulogovan na sajt banke i nije se odjavio, a banka koristi kolačiće za autentifikaciju prehtodni HTML kod (učitavanje slike) će izvšiti neutorizovan prenos novca sa Bobovog računa.

4.5 Ostala tehnička ograničenja kolačića

Pored sigurnosnih ograničenja koja smo opisali u prethodnoj tački, kao i problema koji mogu nastati zbog nepoštovanja privatnosti korisnika, postoje i drugi problemi sa korišćenjem kolačića:

Neprecizna identifikacija

Ukoliko korisnik koristi više pregledača na istom uređaju svaki od njih će posebno da skladišti kolačiće. To znači da kolačići, u opštem slučaju, ne identifikuju osobu koja koristi uređaj već kombinaciju korisničkog naloga, računara i pregledača. Takođe kolačići ne prave razliku između više različitih korisnika koji dele isti korisnički nalog, računar i pregledač.

Nekonzistentno stanje na klijentu i serveru

Korišćenje kolačića može, u određenim slučajevima, dovesti do nekonzistentnog stanja na klijetu i stanja sačuvanog u kolačiću. Ako je kolačić generisan, a zatim korisnik klikne na „back“ dugme stanje pregledača nije isto kao i pre te akcije. Kao primer možemo navesti korpu za kupovinu: ako korisnik doda artikal u korpu a zatim klikne na „back“ dugme stanje korpe se neće promeniti, iako korisnik očekuje da će korpa biti prazna jer je uradio akciju vraćanja unazad. Ovakve greške dovode do nepouzdanosti, konfuzije i softverskih grešaka koje se teško otkrivaju. Zbog toga nije preporučljivo implementirati kompleksna stanja uz pomoć kolačića.

Nekonzistentna podrška na uređajima

Problem u korišćenju kolačića na mobilnim uređajima je da veliki broj uređaja uopšte ne podržava kolačiće. Npr. Nokia podržava kolačiće na 60% svojih uređaja, dok Motorola podržava kolačiće na samo 45% [5]. Pored toga neke mrežne kapije i mobilne mreže u potpunosti uklanjaju kolačiće, dok druge mreže simuliraju kolačiće u ime svojih mobilnih uređaja. Kad je u pitanju prenos kolačića kroz bežične mreže tu postoje velike razlike, tako na primer u Velikoj Britaniji 94% uređaja podržava bežične kolačiće dok ih u SAD podržava svega 47%. U Japanu je situacija dosta bolja obzirom da su bežični uređaji u upotrebi duže vreme, i većina uređaja i mreža podržava kolačiće.

4.6 Alternative za HTTP kolačiće

Neke od funkcionalnosti koje omogućavaju HTTP kolačići mogu se postići i pomoću drugih mehanizama.

IP adrese

Neki korisnici mogu se identifikovati pomoću IP (Internet Protocol) adrese. IP adrese su definisane u mrežnom sloju Internet protokola i koriste se u svim višim slojevima. IP adresa obično je vezana za računar korisnika, osim ako nije u pitanju adresa mrežnog prolaza kada može biti više korisnika iza jedne IP adrese. Prilikom slanja HTTP zahteva server dobija IP adresu i port klijenta, tako da se ova kombinacija u određenim slučajevima može koristiti za identifikaciju sesije korisnika.

Osim što više različitih korisnika može da koristi istu IP adresu postoje i druge manjkavosti ovakvog načina identifikovanja a to su sistemi kao što je Tor pregledač koji su dizajnirani da skrivaju IP adrese korisnika a moguće je uraditi i lažiranje IP adrese. To znači da bi IP adrese trebalo korstiti samo u kombinaciji sa drugim mehanizmima identifikacije.

URL parametri

Ova tehnika se sastoji u dodavanju identifikacije kroz URL parametar. Obično se za ovu svrhu koriste dinamički parametri adrese (query string), mada i drugi delovi URL-a mogu biti korišćeni. Većina serverskih jezika već ima ugrađene mehanizme za ovakav način identifikovanja sesije ukoliko kolačići nisu dostupni. Parametri URL-a su veoma slični kolačićima pošto mogu da sadrže proizvoljne tekstualne informacije.

Postoje ozbiljni nedostaci ovakvog pristupa, a oni se posebno ogledaju u potencijalnim sigurnosnim rizicima zbog deljenja URL-ova. Ukoliko je npr. korisnik ulogovan u svoj nalog i pošalje neki link koji sadrži sesijski parametar drugom korisniku, korisnik koji otvori link može biti automatski ulogovan (preuzimanjem sesije) u nalog prvog korsnika. Drugi slučaj je ako su u parametru enkodirana podešavanja ili druge osetljive informacije – slanjem URL-a automatski dolazi do kompromitacije takvih informacija. Ukoliko se koriste URL parametri preko nekriptovanog kanala postoji mogućnost da napadač koji prisluškuje saobraćaj preuzme sesiju korisnika prostim pregledom i kopiranjem presretnutih zahteva.

Skrivena polja u formama

Ovaj pristup zasniva se na korišćenju skrivenih polja u veb formama za praćenje sesije korisnika. Ukoliko forma koristi HTTP GET metod za slanje informacija na server ova tehnika se ne razlikuje od URL parametara. Ukoliko forma koristi HTTP POST metod onda je situacija nešto drugačija obzirom da se skrivena polja u tom slučaju prenose kroz telo zahteva a ne kroz URL. POST metod je sigurniji u smislu da proksi serveri ne kreširaju i ne loguju POST zahteve. Takođe na ovaj način korisnik ne može slučajno poslati identifikator sesije slanjem URL-a.

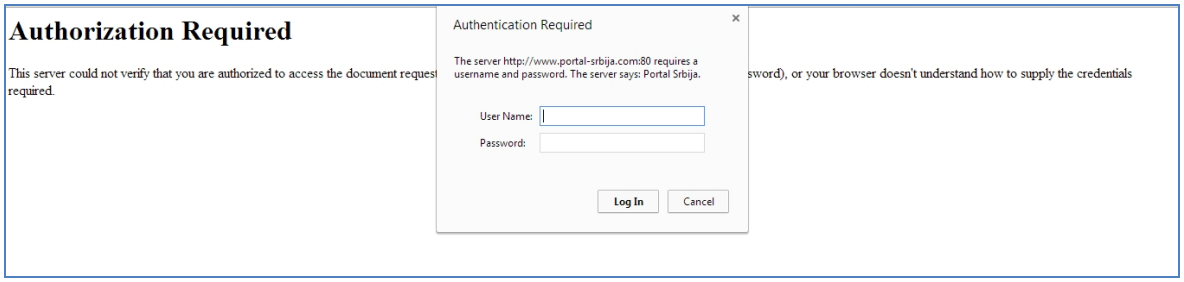
`window.name`

Većina pregledača podržava čuvanje relativno velike količine podataka (2–32 MB) preko JavaScript-a koristeći `window.name` DOM (Document Object Model) osobinu. Ovi podaci mogu se koristiti umesto sesijskih kolačića i pored toga ne zavise od domena. Ova tehnika može se kombinovati sa JSON (JavaScript Object Notation) objektima za čuvanje kompleksnih struktura podataka na strani klijenta.

Nedostatak ovog pristupa je da svaki novi tab inicijalno ima prazan `window.name`, što znači da novootvoreni tabovi inicijalno nemaju naziv prozora. Pored toga `window.name` može biti iskorišćen u maliciozne svrhe obzirom da ne zavisi od domena na kome se korisnik trenutno nalazi.

HTTP autentifikacija

HTTP protokol pruža vrlo jednostavan mehanizam kontrole pristupa na nivou samog protokola i većina veb servera podržava ovaj mehanizam. Ukoliko je veb server podešen da zahteva korisničko ime i lozinku za pristup određenom resursu pregledač prikazuje odgovarajuću formu nakon prijema odgovora sa zahtevom za autentifikaciju:



Slika 3 Forma osnovne HTTP authentifikacije

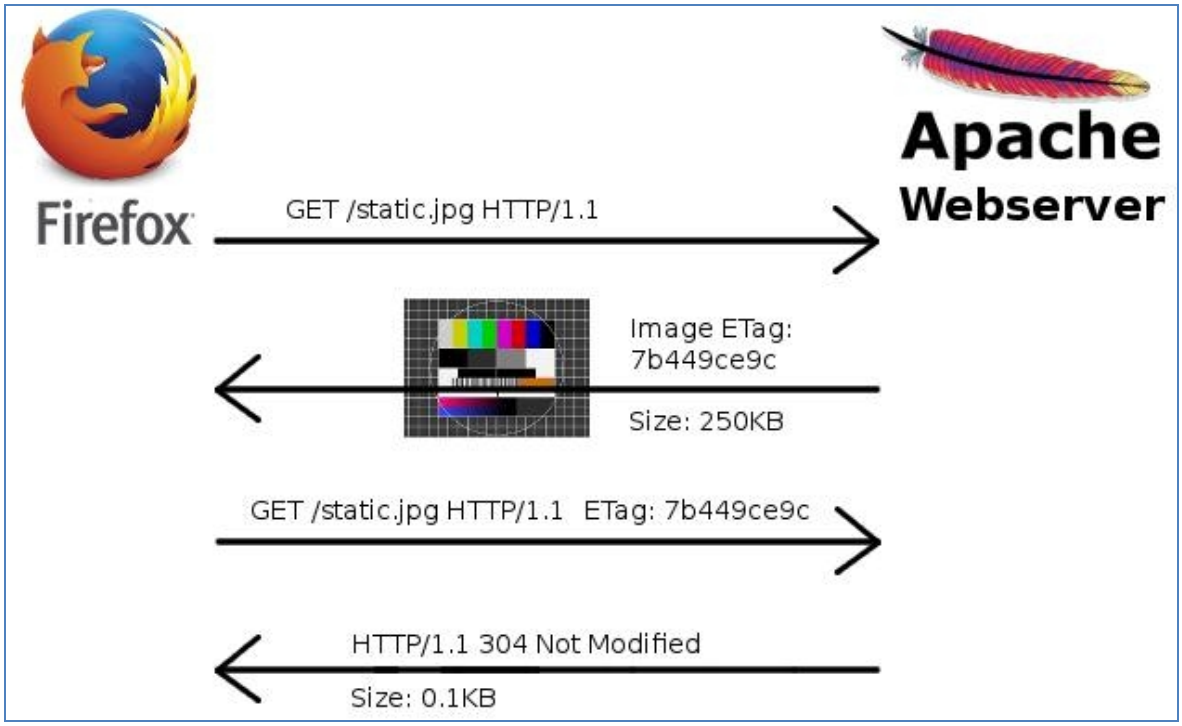
Nakon uspešne autorizacije pregledač čuva korisničko ime i lozinku i šalje ih u narednim zahtevima. Ove informacije mogu biti iskorišćene za praćenje korisnika.

4.7 ETag

Korišćenje Etag-ova umesto kolačića je posebno zanimljiva tehnika obzirom da nema posebnih zahteva koji se tiču klijeta (JavaScript, Flash, Local Storage i sl.). ETag (entity tag) je sastavni deo HTTP protokola i predstavlja jedan od mehanizama validacije keša (browser cache). Pomoću ETag-a klijentima je omogućeno da šalju kondicionalne zahteve. Na ovaj način pregledač postaje efikasniji, smanjuje se količina podataka koja se prenosi, veb server ne mora ponovo da šalje podatke koji se nisu promenili. ETag-ovi se takođe koriste u optimističnoj kontroli konkurentnosti kao način da se spreči simultano ažuriranje resursa od strane više procesa.

Tehnički gledano ETag predstavlja nevidljiv identifikator (odnosno vidljiv samo u HTTP zaglavlju) specifične verzije resursa pronađenog na određenom URL-u. Ako se resurs promeni automatski se menja i ETag. ETag je implementiran tako da se brzo generiše i poredi. Poređenje ETag-ova ima smisla samo za isti URL obzirom da ETagovi za različite URL-ove mogu i ne moraju biti isti ali se iz toga ništa ne može zaključiti.

Iako ne toliko rasprostranjena tehnika praćenja (identifikovanja) korisnika pomoću ETag-ova je podjednako efikasna (u nekim slučajevima efikasnija) kao i ostale. Ova tehnika zasniva slanju HTTP zahteva za proveru da li postoji nova verzija keširanog resursa (slika 4) [6]:



Slika 4 Provera nove verzije keširane slike

Ako bolje pogledamo videćemo da se radi se sličnom konceptu kao i kod HTTP kolačića – klijent ponovo šalje serveru informaciju koju je već primio. Dakle ostaje samo da server koji želi da prati klijente ETag prilagodi klijentu (npr. `Hash(IP adresa + User-agent string)`) (kod u programskom jeziku PHP):

```
// ETag je poslat na server
if (!empty($_SERVER["HTTP_IF_NONE_MATCH"])) {
  $etag = substr(str_replace(".", "", str_replace("/", "", str_replace("\\", "",
$_SERVER["HTTP_IF_NONE_MATCH"]))), 0, 18);
} else { // ETag nije detektovan, i zato generišemo novi
  $etag = substr(sha1($secret . sha1($_SERVER["REMOTE_ADDR"]) .
sha1($_SERVER["HTTP_USER_AGENT"])), 0, 18);
}

//Slanje zaglavlja
header("Cache-Control: private, must-revalidate, proxy-revalidate");
header("ETag: " . substr($etag, 0, 18));
header("Content-type: image/jpeg");
header("Content-length: " . filesize("fingerprinting.jpg"));
```

5. Metodologija kreiranja digitalnog otiska prista (fingerprinting)

Digitalni otisak prsta (fingerprint) uređaja ili identifikator uređaja je skup sistemskih atributa koji, za svaki posmatrani uređaj, imaju skup odgovarajućih vrednosti, koje zajedno uzete čine (sa velikom verovatnoćom jedinstvenosti) identifikator uređaja. Ovi atributi uključuju, npr. veličinu ekrana, verzije instaliranog softvera, listu instaliranih fontova, osobine hardvera na uređaju. Atributi koji imaju više različitih vrednosti (npr. lista instaliranih fontova) više učestvuju u konačnoj identifikaciji. Slično tome, atributi koji

imaju tendenciju da su stabilni sa protokom vremena takođe više doprinose identifikaciji.

Identifikacija klijenata u veb aplikacijama je proces prikupljanja i čuvanja potrebnih informacija o pregledačima, bez snimanja stanja na strani klijenta. Ovako dobijeni identifikatori mogu služiti za praćenje klijenata na vebu, ali imaju i druge primene kao što je pojačavanje sigurnosti standardnih metoda autentifikacije i prenosa sesije ili otkrivanje malicioznih aktivnosti. U zavisnosti od potrebe aplikacije identifikacija uređaja se može vršiti različitim algoritmima, ali se suštinski svodi na odabir određenog podskupa atributa. Npr. ukoliko je potrebno proveriti da li aplikacija podržava Flash ili HTML5 radi isporučivanja multimedijalnog sadržaja relativno mali broj atributa je potreban da bi se to utvrdilo.

5.1 Metode bazirane na JavaScript-u

Od svog nastanka 1995. godine do danas JavaScript se razvio u defakto standard za programiranje na strani klijenta. Prisutnost JavaScript-a u pregledačima kao i veliki biblioteka otvorenog koda čine JavaScript nezamenljivim jezikom za razvoj veb aplikacija. Prema podacima[1] koji su nam bili dostupni u toku pisanja ovog rada svega 0.74% klijenata nema uključen JavaScript, dok je taj broj na globalnom nivou, prema podacima koje je Yahoo objavio 2010 [7], oko 1%.

JavaScript se obično koristi za manipulaciju DOM drvetom, izvršavanje asinhronih zahteva, i dodavanje interaktivnosti elementima korisničkog interfejsa. JavaScript ima privilegovanu poziciju unutar pregledača obzirom da može pristupiti većini interfejsa i resursa koje pregledač pruža. Resursi pregledača kojima se može pristupiti iz JavaScript-a, a koji su od značaja za identifikovanje klijenta su:

- `navigator` – objekat koji sadrži informacije o proizvođaču pregledača, verziji pregledača, podržanim dodacima, MIME (Multipurpose Internet Mail Extensions) tipovima kao i detaljne informacije o operativnom sistemu i arhirekturi na kojoj se pregledač izvršava
- `screen` – je objekat koji sadrži informacije o ekranu korisnika (širinu i visinu, dubinu boja i gustinu piksela)

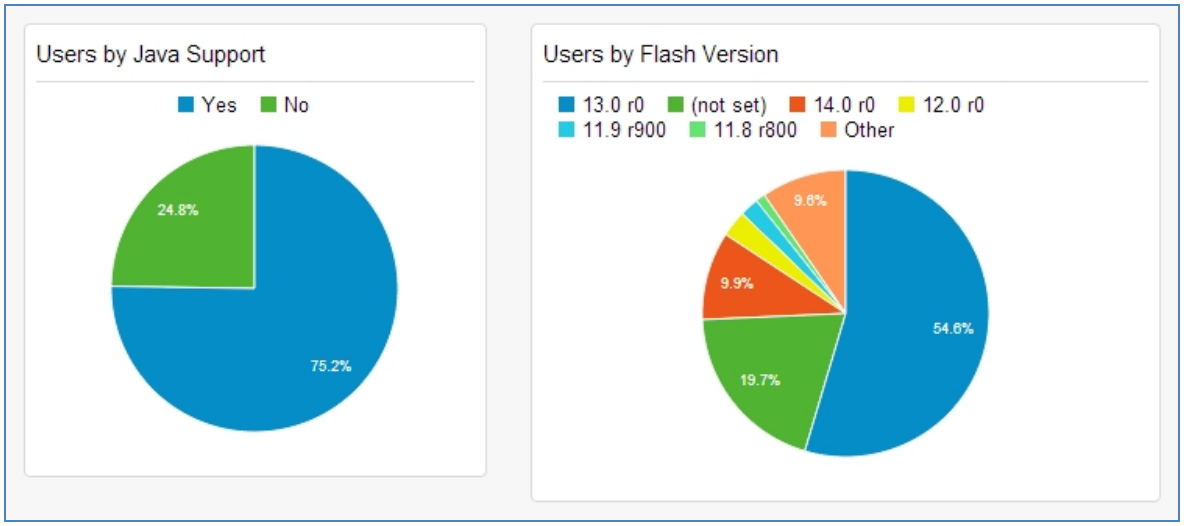
U eksperimentu [8] iz 2009. Majer je pokazao da je kombinacijom atributa ova dva objekta uspeo da napravi identifikator koji uspešno identifikuje 96% pregledača. Godinu dana kasnije Ekersli je u već pomenutom Panopticlick projektu unapredio algoritam i ispitao oko 500.000 pregledača, pri čemu je jedinstveno identifikovao 94.2%. Ekersli je takođe pokazao da je lista fontova instaliranih na računaru jedan od faktora koji najviše doprinosi jedinstvenosti identifikatora. Lista fontova može se dobiti korišćenjem JavaScript-a merenjem i poređenjem dimenzija teksta iscrtanog različitim fontovima [9] ili uz pomoć kombinacije JavaScripta-a i Flash dodatka.

Ostale tehnike koje se baziraju na JavaScript-u su korišćenje merenja performansi JavaScript platforme, grešaka u standardnim testovima i razlike u prikazu `canvas` elementa koji je programski kontrolisan iz JavaScript-a. Pored ovoga istorija pregledanja stranica koja se može rekonstruisati uz pomoć boje posećenog linka, takođe može biti iskorišćena za pravljenje identifikatora. Nije poznato da li su neke od ovih tehnika ispitane na statistički značajnim uzorcima. U našoj implementaciji koristićemo razlike u prikazu `canvas` elementa i pokazaćemo kakav je uticaj ove tehnike na jedinstvenost identifikatora.

[1] Pristupni logovi na sajtu www.portal-srbija.com, slučajni uzorak od 70.000 posetilaca

5.2 Metode bazirane na dodacima pregledača

Poslednja verzija HTML standarda, HTML5, proširuje mogućnosti JavaScript-a i CSS-a (Cascading Style Sheets) i omogućava programerima da kreiraju kompleksne aplikacije, po mogućnostima veoma bliske desktop aplikacijama. U prethodnim verzijama HTML-a to nije bilo moguće (npr. igre, video i audio sadržaji su bili ograničeni u starijim pregledačima). Zato su razne kompanije razvijale dodatke za pregledače, obično da bi mogao da se prikaže multimedijalni sadržaj. Najznačajniji dodaci, koji su i danas prisutni u pregledačima su svakako Adobe Flash i Java. Prema podacima koji su nam dostupni ovi dodaci pokazuju tendenciju ka manjoj upotrebi. Trenutno[2] 19.7% korisnika nema Flash dodatak, dok 24.8% korisnika nema instaliran Java dodatak.



Slika 5 Podrška za Java i Flash dodatke

Kao i kod JavaScript-a, široka prihvaćenost pomenutih dodataka omogućava lakše i preciznije kreiranje identifikatora. Ekersli je u svojoj implementaciji koristio Flash i Java dodatke za preuzimanje liste fontova instaliranih na računaru. Pored toga poznato je da se Flash može iskoristiti da se zaobiđe proksi definisan od strane korisnika i na taj način dobije više informacija o uređaju korisnika (verzija jezgra operativnog sistema i prisustvo više monitora).

Proširenja pregledača (browser extensions), takođe su od značaja za identifikovanje ali njima se ne može pristupiti iz JavaScript-a, tako da ih je moguće detektovati samo kroz bočne efekte (ako postoje). Moveri [10] je pokazao da je moguće doći do korisničke liste omogućenih sajtova za popularno NoScript proširenje, tako što se zahtevaju skriptovi sa različitih domena a zatim se proverava da li su se izvršili preko određenih objekata koji se nalaze u globalnom prostoru promenljivih u slučaju uspešnog izvršavanja. Ovako dobijena lista može se iskoristiti kao dodatni parametar identifikacije.

Proširenja koja se koriste za lažiranje korisničkog agenta takođe mogu biti otkrivena zbog nekonzistentnosti u pregledaču kada su takve ekstenzije aktivne.

5.3 Metode bazirane na HTTP zaglavljima i serverske metode

Studija o identifikaciji [11], slična Ekerslijevoj, u kojoj su analizirani pristupni logovi Bing-a i Hotmail-a, pokazala je da je IP adresa u kombinaciji sa identifikatorom

[2] Google Analytics za www.portal-srbija.com, podaci za maj i jun 2014., uzorak od 280.000 korisnika

korisničkog agenta dovoljna za pravljenje preciznih identifikatora za praćenje.

Pre nego što je identifikovanje bilo moguće na nivou pregledača istrazivači su pokazali je da je moguće udaljenim putem otkriti operativni sistem korisnika. Takođe, moguće je identifikovati korisnike koji se nalaze iza NAT-a (Network Address Translation) preko razlika u njihovim časovnicima, analizirajući vremena unutar TCP paketa [12]. Iako ove tehnike nisu toliko precizne kao one na nivou pregledača, činjenica da se izvršavaju samo na serveru govori nam da je njihova detekcija teška, ako ne i nemoguća.

5.4 Identifikacija na osnovu liste instaliranih fontova

Važnu ulogu u našoj implementaciji identifikacije klijenata ima i tehnika preuzimanja liste instaliranih fontova. U FPDetective projektu [13], u kome je vršeno pronalaženje skriptova koji vrše identifikaciju (suštinski, inverzan proces onome kojim se ovaj rad bavi) opisan je eksperiment kombinovanog automatskog prepoznavanja i ručnog sortiranja većine poznatih tehnika identifikacije. Na taj način je pokazano da je preuzimanje liste instaliranih fontova vrlo važan sastojak u identifikaciji uređaja. Štaviše prisustvo poziva za dobijanje liste fontova je siguran znak da takav skript vrši identifikaciju klijenta, obzirom da takva akcija ima veoma malo drugih potencijalnih namena.

Korišćenje fontova za identifikovanje klijenata ima i druge prednosti:

- prema Panopticlick studiji lista fontova je drugi najvažniji atribut identifikacije uređaja sa 17.1 bitova entropije (nesortirano)
- iako lista instaliranih dodataka ima nešto više entropije, lista fontova zavisi samo od operativnog sistema tako da može da služi za povezivanje različitih pregledača na istom uređaju
- Tor pregledač, koji je najraširenij alat za odbranu od identifikovanja, instalira se sa predefinisanom konfiguracijom, u kojoj je instaliranje dodataka onemogućeno. Fontovi su, dakle, najvažniji za identifikaciju u ovom kontekstu i verovatno jedini atribut za uspešno identifikovanje anonimnih korisnika Tor pregledača

Kao što je već pomenuto (u 5.1) lista fontova može se dobiti uz pomoć Flash-a (odnosno ActionScripta koji se kompajlira u Flash format) i JavaScript-a. Ove metode se razlikuju u načinu pristupa obzirom da JavaScript ne može neposredno da pročita listu fontova. U slučaju korišćenja Flash-a dinamička detekcija identifikacije bazira se na dekompajliranju Flash bajt koda (byte code), odnosno formata koji izvršava Flash interpreter koji je sastavni deo Flash dodatka za pregledač. Rezultat dekompajliranja je ActionScript kod na kome se lako može izvršiti statička analiza i pronaći pozivi za enumeraciju fontova. Takođe, analizom ActionScript koda može se utvrditi da li se ispitivanje fontova koristi u neke druge (legitimne) svrhe (npr. za prikaz teksta u multimedijalnim sadržajima).

Sa druge strane, zbog sve prisutnije upotrebe minimizacije koda i drugih tehnika namernog smanjivanja čitljivosti, uopšteno je veoma teško izvršiti statičku analizu JavaScript koda. Zato je za otkrivanje tehnika identifikacije bolje koristiti dinamičke metode presretanja poziva koji rade otkrivanje fontova. Na sličan način se otkrivaju i drugi načini identifikovanja (provera osobina pregledača: `name, filename, description, length, userAgent, appCodeName, product, and productSub`, provera instaliranih dodataka: `enabledPlugin, description, suffixes, type`, provera

osobina ekrana i provera atributa koji pokazuju da li su instalirane trake sa pomoćnim alatima: `availTop, availLeft`).

JavaScript kod se može klasifikovati kao identifikator ako učitava više od 30 sistemskih fontova, ako izlistava dodatke i MIME tipove, ako proverava osobine ekrana i pregledača i sve pomenute podatke šalje na server.

5.5 Identifikacija na osnovu HTML5 osobina

HTML5 donosi velika poboljšanja i nove koncepte u razvoj aplikacija na strani klijenta. Uopšteno, HTML5 ima sledeće ciljeve:

- Primarni cilj je svakako konzistentna, dobro definisana obrada grešaka. Kao što je poznato HTML interpreteri su napravljeni tako da ignorišu i prepravljaju greške u HTML dokumentima i na taj način formiraju validne dokumente. Problem je što ne postoje pravila koje greške su "dobre" a koje "loše". Ukoliko neko želi da napravi veb pregledač on mora da testira nepravilno formirane dokumente u raznim pregledačima (a posebno u Internet Explorer-u) i da radi reverzni inženjering njihove obrade grešaka. U suprotnom, mnoge stranice se neće prikazati korektno (procene govore da je približno 90% stranica na Internetu na neki način loše formirano). Zato HTML5 pokušava da otkrije i kodifikuje obradu grešaka, tako da programeri klijentskih veb aplikacija mogu da rade po standardima i na taj način drastično smanje vreme i novac potreban da bi aplikacije radile korektno. Takođe, ukoliko u budućnosti HTML format bude odbačen ili zamenjen nečim drugim, onima koji budu hteli da čitaju naše HTML dokumente biće mnogo lakše ukoliko imaju kompletno definisan algoritam parsiranja
- Drugi cilj je da se veb pregledač razvija kao platforma za aplikacije preko HTML-a, CSS-a i JavaScript-a. Mnogi elementi, koji su u HTML4 bili mogući samo uz pomoć Flash-a ili nekih programskih trikova, su dodati direktno u standard (`<canvas>, <video>, <audio>`). Korisne tehnologije kao što je lokalno skladištenje (Local Storage), novi tipovi polja za unos, i validacija formi na nivou pregledača, čine razvoj aplikacija mnogo jednostavnijim za programere i bržim za krajnje korisnike.
- Niz manjih poboljšanja kao što su bolje definisana semantička uloga postojećih elemenata (`<strong>, <em>`), dodavanje potpuno novih elemenata (`<article>, <section>, <header>, <aside>` i `<nav>`) koji će značajno olakšati pisanje i pregled koda na veb stranama

Ostale važne razlike u odnosu na HTML4:

- Parsiranje nije bazirano na SGML (Standard Generalized Markup Language) već na novim pravilima parsiranja koja su fleksibilna, ali zadržavaju i kompatibilnost sa prethodnim standardima
- Mogućnost korišćenja SVG (Scalable Vector Graphics) i MathML direktno u HTML-u
- Novi elementi: `article, aside, audio, bdi, canvas, command, data, datalist, details, embed, figcaption, figure, footer, header, keygen, mark, meter, nav, output, progress, rp, rt, ruby, section, source, summary, time, track, video, wbr`
- Novi tipovi kontrola u formama: vreme i datum, email, url, pretraga, broj, opseg, telefon, odabir boje

- Novi atributi: `charset` (označava kodni raspored), `async` (za odloženo učitavanje skriptova)
- Globalni atributi koji mogu da se primene na bilo koji element: `id, tabindex, hidden, data-*` (podesivi data atributi)
- Zastareli elementi su kompletno izbačeni: `acronym, applet, basefont, big, center, dir, font, frame, frameset, isindex, noframes, strike, tt`

Novi API (Application Programming Interface) koje omogućava HTML5 su:

- Element Canvas sa 2D crtanjem
- Offline veb aplikacije
- Izmena dokumenata
- Vuci-i-baci (Drag-and-drop)
- Slanje poruka između dokumenata
- Upravljanje istorijom pregledača
- Registrovanje MIME tipova i protokola
- Mikro formati (schema.org i drugi)
- Veb skladištenje – okvir za skladištenje ključ-vrednost parova koji omogućava funkcionalnost sličnu kolačićima ali sa više kapaciteta za skladište i unapređenim API-em

Nisu sve nabrojane tehnologije deo W3C HTML5 standarda iako se nalaze u dokumentaciji radne grupe za HTML standardizaciju. Druge, povezane tehnologije W3C publikuje zasebno, mada ih većina savremenih pregledača već podržava:

- Geolociranje
- Indeksirane baze
- HTML5 pristup fajlovima i prenos fajova na server
- Pisanje fajlova – API za upis u lokalne fajlove iz pregledača
- Veb Audio API, JavaScript API visokog nivoa za procesiranje i sintetisanje zvuka u veb aplikacijama

Za problem identifikacije klijenata svakako je najznačajniji `canvas` element kao i odgovarajući JavaScript API. `Canvas` je element koji omogućava da se unapred na definisanom regionu na veb strani može programski crtati. Podrška u pregledačima za ovaj element je više nego dobra – `canvas` podržavaju svi desktop pregledači (Chrome, Firefox, Opera, Safari) kao i mobilni Chrome i mobilni Safari.

U osnovi programski pristup `canvas` elementu je vrlo jednostavan: instanciranje grafičkog konteksta (objekta) i korišćenje dostupnih metoda za pravljenje željenih izmena. U trenutnoj HTML5 specifikaciji jedini definisani kontekst je "2d" koji podržava osnovne metode za crtanje kao što su `fillRect, lineTo, arc`, kao i komplikovanije funkcije kao što su Bezijeove krive, gradijenti i kopiranje postojeće slike.

Ispis teksta je takođe jedna od funkcija 2d konteksta kanvas elementa. Ukoliko se prosledi veličina fonta, vrsta fonta i osnovna linija 2d kontekst može da ispiše tekst. Ukoliko širina teksta prelazi širinu elementa tekst će biti "odsečen" na ivici elementa. Nad iscrtanim tekstom mogu se primeniti stilovi slični CSS-u. Primer opisanog procesa u JavaScript-u:

```
<script type="text/javascript">
var canvas = document.getElementById("drawing");
var context = canvas.getContext("2d");
context.font = "18pt Arial";
context.textBaseline = "top";
context.fillText("Hello, user.", 2, 2);
</script>
```

Da bi `canvas` mogao da se upotrebi za pravljenje identifikatora potrebno je pronaći način da se ispita njegovo ponašanje. Za tu svrhu `canvas` pruža nekoliko načina da se ispitaju njegovi podaci na nivou piksela. U 2d kontestu `canvas` poseduje metod `getImageData()`. Za prosleđeni pravougaoni region `canvas`-a ovaj metod vraća `ImageData` objekat koji sadrži RGBA vrednosti (celobrojne) svakog piksela u zahtevanom regionu.

Pored toga sam `canvas` objekat poseduje `toDataURL(type)` metod. Kada se ovom metodu prosledi "`image-png`" on vraća sliku u PNG formatu u Base64 enkodiranoj formi, u kojoj se nalazu celokupan sadržaj `canvas`-a. Ovaj metod je najpodesniji za preuzimanje podataka iz `canvas`-a radi dalje analize. U slučaju pravljenja indentifikatora dovoljno je uraditi heš (hash, kontrolna suma) obzirom da se za ove svrhe može smatrati da su većina heš funkcija obostrano jednoznačna preslikavanja. Ovim se značajno smanjuje opterećenje i prostor potreban za istraživanje identifikatora pogotovo u slučaju sajtova sa velikim brojem korisnika. Važno je napomenuti da ovi metodi zadržavaju polise istog izvora (ne može se ubaciti npr. slika koja nije sa istog domena na kome je i kreiran `canvas`), što je veoma važno za identifikovanje zato što klijent ne može da ubaci proizvoljnu sliku.

Veb fontovi, definisani u CSS3, dozvoljavaju programerima da učitavaju fontove po potrebi sa proizvoljnih veb lokacija, umesto da koriste fontove instalirane na računaru klijenta. Za uključivanje fonta programer koristi CSS direktivu `@font-face` sa odgovarajućim `src` atributom koji pokazuje na adresu fonta na Internetu. Prilikom učitavanja veb strane pregledač preuzima font sa zadate lokacije i omogućava njegovo korišćenje na veb strani. Veb fontovi se mogu koristiti i prilikom ispisivanja teksta na `canvas` elementu što dodatno doprinosi upotrebnoj vrednosti `canvas`-a za identifikovanje. Za učitavanje veb fontova najčešće se koriste Google ili Typekit biblioteke. Sa ovim bibliotekama veb fontovi se mogu učitati samo uz pomoć JavaScript-a, a moguće je i definisati akcije za slučaj da font jeste odnosno nije uspešno učitan. Identifikacija se obavlja unutar definisane akcije za slučaj uspešnog učitavanja fonta.

Nove mogućnosti koje smo prethodno opisali, uključujući i WebGL (Web Graphics Library) koji se takođe može koristiti za identifikovanje (ali to izlazi van okvira ovog rada), imaju i neke ozbiljne sigurnosne implikacije. Zbog većih potreba za brzinom izvršavanja veb skriptovi imaju mogućnost da pristupe pozivima na nivou jezgra operativnog sistema. Maliciozni skriptovi mogu da uzrokuju zaustavljanje celog operativnog sistema i da resetuju grafičke kartice na Windows-u. Veb fontovi takođe mogu biti uzrok ranjivosti jer su već pronađene tehnike izvršavanja udaljenog koda prilikom parsiranja TrueType fontova, nezavisno od operativnog sistema. Iako ove sigurnosne implikacije nisu u direktnoj vezi sa tehnikama identifikacije (u smislu da tehnike identifikacije nisu usmerene niti prouzrokuju štetu na računaru korisnika), ipak ove tehnike koriste činjenicu da potencijalno nesiguran sadržaj dospeva do sistemskih poziva niskog nivoa.

6. Implementacija identifikacije klijenata

6.1 Algoritam identifikacije

Algoritam identifikacije u pregledaču sastoji se od prikupljanja karakteristika nabrojanih u tački 5. Neki od ovih parametra dobijaju se pristupanjem odgovarajućim osobinama objekata u JavaScript-u, a drugi se dobijaju kao rezultati izračunavanja funkcija koje smo napisali. Ukupno smo koristili 17 različitih parametara a krajnji rezultat (identifikator) dobija se nadovezivanjem parametara a zatim primenom heš funkcije. Svi parametri sa kratkim opisom nalaze se u tabeli:

Promenljiva	Opis
`User Agent`	sadrži mikro-verziju browsera, verziju operativnog sistema, jezik operativnog sistema, intalirane toolbar-ove i druge informacije
`Language`	oznaka jezika u pregledaču, nema poseban format
`Cookies enabled`	Ima vrednost No ako su kolačići onemogućeni
`Screen color depth`	broj bitova korišćen da se označi boja u jednom pikselu ili komponenta boje u jednom pixelu
`Screen resolution`	rezolucija ekrana u pikselima (HxW)
`Timezone offset`	razlika vremenske zone klijenta u odnosu na UTC u minutima
`Session storage`	da li pregledač podržava čuvanje sesija (true/false)
`Local storage`	da li pregledač podržava lokalno skladištenje podataka
`IdenxedDB`	da li pregledač podržava indeksirane baze
`Body addBehavior`	da li pregledač podržava addBehavior metod (karakteristično za Internet Explorer)
`Open database`	da li pregledač podržava open database tehnologiju
`CPU class`	klasa procesora na računaru na kome se pregledač izvršava
`Platform`	operativni sistem na kome se pregledač izvršava
`Do not track`	da li je uključena “do not track” opcija u pregledaču
`Plugins hash`	lista dodataka instalirana na računaru
`Fonts`	lista fontova instalirana na računaru
`Canvas`	kanvas identifikator, kao što je opisano u 5.4

Tabela 1 Parametri za identifikaciju

U većini slučajeva pristupamo `navigator`, `window`, `screen` i `body` objektima u JavaScript-u. Izuzetak su lista dodataka (plugins), lista fontova i `canvas` identifikator. Ukoliko je JavaScript onemogućen onda ćemo dobiti podrazmevane vrednosti (prazan string) za pomenute izuzetke, tako da će kompletna procedura identifikacije biti neuspešna.

Obzirom na 5.2 naša implementacija ne sadrži ni parametre zasnovane na prisustvu Java dodatka i rezultatima funkcija koje se u njemu izvršavaju. Trudili smo se da se ograničimo na tehnike koje su najšire podržane da bismo dobili što bolju sliku uspešnosti

identifikovanja. Svi podaci su prikupljeni potpuno anonimno i poslati na server Ajax zahtevom. Iako je ceo proces urađen bez znanja korisnika nema opasnosti da će doći do ugrožavanja privatnosti zato što nisu zabeleženi podaci o ličnosti ili nalozima korisnika na bilo kom servisu i zato što je u pitanju implementacija za nekomercijalne (akademske) svrhe.

6.2 Matematička osnova

Pretpostavimo da imamo algoritam identifikacije $F(\cdot)$, takav da kada se primeni na nove instalacije pregledača x, rezultat $F(x)$ prati diskretnu raspodelu verovatnoća $P(f_n)$, $n \in [0,1,\dots N]$. Sopstvena informacija ili „faktor iznenađenja“ određenog algoritma data je sa:

$$I(F(x) = f_n) = -log_2(P(f_n))$$

Faktor iznenađenja I meren je u bitovima što je rezultat izbora broja 2 za bazu logaritma. Entropija distribucije $P(f_n)$ je očekivana vrednost sopstvene informacije za sve posmatrane pregledače i data je sa:

$$H(F) = -\sum_{n=0}^{N} P(f_n) log_2(P(f_n))$$

Sopstvena informacija se može definisati i kao količina informacija o identitetu objekta koji se identifikuje, gde svaki bit informacija smanjuje broj mogućnosti za polovinu. Ako je sajt posećen, sa jednakom verovatnoćom, iz skupa od X različitih pregledača, intuitivno možemo pretpostaviti da će određeni pregledač $x \in X$ biti jedinstveno identifikovan ako je

$$I\big(F(x)\big) \gtrapprox log_2|X|$$

Ova intuitivna pretpostavka može se zameniti binomnom raspodela sa odgovarajućim intervalima pouzdanosti.

U slučaju da je identifikator formiran kombinovanjem više različitih komponenata $F_s(\cdot)$, $s \in S$, onda ima smisla govoriti o sopstvenoj informaciji svake od komponenata i definisati odgovarajuću entropiju komponente identifikatora:

$$I_s\big(f_{n,s}\big) = -log_2\big(P\big(f_{n,s}\big)\big)$$

$$H_s(F_s) = -\sum_{n=0}^{N} P\big(f_{s,n}\big) log_2\big(P\big(f_{s,n}\big)\big)$$

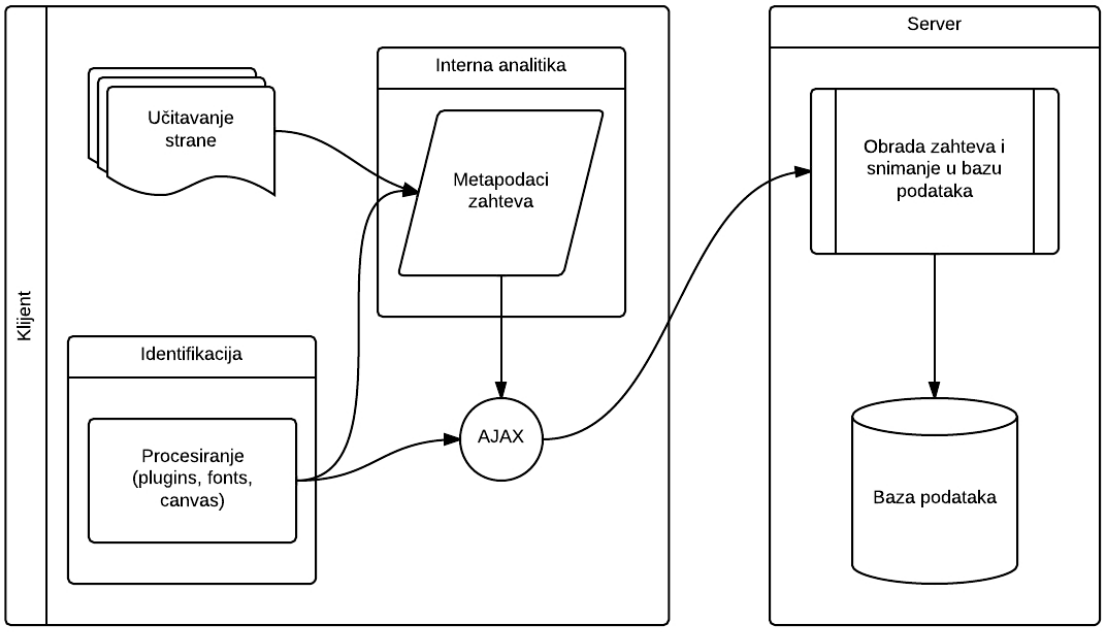
Primetimo da sopstvene informacije dve komponente identifikatora F_s i F_t mogu da se dodaju linearno samo u slučaju da su te dve promenljive statistički nezavisne, što nije slučaj u praksi. Umesto toga koristimo kondicionalnu sopstvenu informaciju:

$$I_{s+t}(f_{n,s}, f_{n,t}) = -log_2(P(f_{n,s}|f_{n,s}))$$

Slučajevi kao što je identifikacija blokade Flash-a posebnim ispitivanjem dodatka i fontova su takođe obuhvaćeni jer je $P(fonts = \text{"not detected"} \mid \text{"Flash"} \in plugins)$ jako mala.

6.3 Arhitektura rešenja

Implementacija rešenja za identifikaciju oslanja se na već postojeću infrastrukturu sajta www.portal-srbija.com (pre svega što se tiče statističke analize efikasnosti implementacije, kao i biblioteka koje su korišćene). Diagram modula i procesa prikazan je na slici 6:



Slika 6 Diagram implementacije identifikacije klijenata

Prilikom implementacije korišćene su biblioteke JQuery (standardna biblioteka za manipulaciju DOM objektima i događajima), JQuery.cookie (dodatak za rad sa kolačićima), Sha1.js (implementacija Sha1 heš algoritma za JavaScript). Centralna biblioteka (Fingerprint) je modifikovana biblioteka otvorenog koda [14].

Kao što se vidi na slici 6, nakon učitavanja strane prvo se prikupljaju meta-podaci zahteva i obzirom da nije potrebno dodatno procesiranje ili čekanje na asinhrone događaje oni se Ajax zahtevom odmah šalju na server. U slučaju da JavaScript nije uključen ili se radi o veb robotu, postoji alternativa u vidu slanja zahteva za slikom sa dodatim parametrima (tzv. web bug tehnika). Na taj način osiguravamo da je zabeležena akcija posetioca. Istovremeno sa ovim podacima prosleđujemo i identifikator bez učešća liste

fontova kao komponente (iz razloga što bi trebalo da čekamo na asinhroni događaj učitavanja Flash-a i liste fontova, što u slučaju naše interne primene nije od značaja).

Nakon male zadrške vršimo prikupljanje podataka koji su značajni za naš eksperiment. Svi podaci osim liste fontova, liste dodataka i kanvas identifikatora šalju se u izvornom obliku. Nabrojani izuzeci šalju se nakon primene heš funkcije (vidi 5.5).

Najvažnija funkcija Fingerprint biblioteke je `get()`:

```
get: function(){
      var keys = [];
      var log_keys_data = [];
      keys.push('navigator_user_agent='+navigator.userAgent);
      keys.push('navigator_language='+navigator.language);
      keys.push('screen_color_depth='+screen.colorDepth);
      if (this.screen_resolution) {
        var resolution = this.getScreenResolution();
        if (typeof resolution !== 'undefined') {
keys.push('screen_resolution='+this.getScreenResolution().join('x'));
        } else {
          keys.push('screen_resolution=');
        }
      } else {
         keys.push('screen_resolution=');
      }
      keys.push('timezone_offset='+new Date().getTimezoneOffset());
      keys.push('has_session_storage='+this.hasSessionStorage());
      keys.push('has_local_storage='+this.hasLocalStorage());
      keys.push('indexed_db='+!!window.indexedDB);
      if(document.body){
keys.push('body_add_behavior='+typeof(document.body.addBehavior));
      } else {
        keys.push('body_add_behavior='+typeof undefined);
      }
      keys.push('window_open_database='+typeof(window.openDatabase));
      keys.push('navigator_cpuclass='+navigator.cpuClass);
      keys.push('navigator_platform='+navigator.platform);
      keys.push('navigator_donottrack='+navigator.doNotTrack);
      keys.push('plugins_string='+Sha1.hash(this.getPluginsString()));
      if(this.font_detect) {
        keys.push('fonts='+Sha1.hash(this.getFonts()));
      } else {
        keys.push('fonts=');
      }

      if(this.canvas && this.isCanvasSupported()){
        keys.push('canvas_fp='+Sha1.hash(this.getCanvasFingerprint()));
      } else {
        keys.push('canvas_fp=');
      }
      var str_keys = keys.join('###');
      var fp = Sha1.hash(str_keys);
      if(this.log_keys) {
            $.ajax({
              type: 'POST',
              url: '/pst.php',
              data: { act: 'logkeys', keys: str_keys, fp: fp  },
```

```
                success: function(resp) {
                return true;
            }
        });
    }

    return fp;
}
```

U ovoj funkciji definišemo niz `keys` koji zatim popunjavamo parovima ključ=vrednost. Prilikom popunjavanja, u nekim slučajevima umesto čitanja osobine objekta pozivamo metode objekta `Fingerprint` (obično je potrebno da se procesiraju dodatne informacije ili postoji obrada koja zavisi od tipa pregledača).

Od ostalih (netrivijalnih) metoda navešćemo metod za listanje fontova uz pomoć Flash-a, metod za listanje dodataka kao i metod za dobijanje `canvas` identifikatora. Metod za listanje fontova sastoji se iz dve komponente – Flash i Javascript.

Flash komponenta [15]:

```
package {

    import flash.display.Sprite;
    import flash.text.Font;
    import flash.text.FontType;
    import flash.text.FontStyle;
    import flash.external.ExternalInterface;

    public class FontList extends Sprite
    {
        public function userFonts():Array
        {
            var embeddedAndDeviceFonts:Array = Font.enumerateFonts(true);

            var deviceFontNames:Array = [];
            for each (var font:Font in embeddedAndDeviceFonts)
            {
                if (font.fontType == FontType.EMBEDDED
                    || font.fontStyle != FontStyle.REGULAR
                    )
                    continue;
                deviceFontNames.push(font.fontName);
            }

            deviceFontNames.sort();
            return deviceFontNames;
        }
        private function addCallbacks() : void
        {
            if(ExternalInterface.available)
            {
                ExternalInterface.addCallback("getUserFonts", userFonts);
            }
        }
    }
```

```
}
```

JavaScript komponenta:

```
getFonts: function() {
  var fonts = "";
  try {
    var obj = document.getElementById("flashfontshelper");
    if (obj && typeof(obj.getUserFonts) != "undefined") {
      fonts = obj.getUserFonts();
    }
    if ("" == fonts)
      fonts = "no_fonts_detected";
    } catch(e) {}
      return fonts;
    }
}
```

Kao što se može videti iz izvornog koda važno je da Flash fajl bude učitan pre poziva iz JavaScript-a. Flash (ActionScript) komponenta koristi standardne pakete `Font`, `FontType` i `FontStyle`, a lista fontova se dobija pozivom `Font.enumerateFonts()`. „Ugnježdeni“ ili fontovi koji nisu regularni se ne uzimaju u obzir.

U JavaScript komponenti vršimo jednostavno povezivanje sa Flash komponentom, a zatim pozivamo metod `getUserFonts()` koji preuzima listu fontova iz Flash komponente.

Preuzimanje liste dodataka:

```
getRegularPluginsString: function () {
   return this.map(navigator.plugins, function (p) {
     var mimeTypes = this.map(p, function(mt){
       return [mt.type, mt.suffixes].join('~');
     }).join(',');
     return [p.name, p.description, mimeTypes].join('::');
   }, this).join(';');
 },

 getIEPluginsString: function () {
   if(window.ActiveXObject){
     var names = ['ShockwaveFlash.ShockwaveFlash',//flash plugin
       'AcroPDF.PDF', // Adobe PDF reader 7+
       'PDF.PdfCtrl', // Adobe PDF reader 6 and earlier, brrr
       'QuickTime.QuickTime', // QuickTime
       // 5 versions of real players
       'rmocx.RealPlayer G2 Control',
       'rmocx.RealPlayer G2 Control.1',
       'RealPlayer.RealPlayer(tm) ActiveX Control (32-bit)',
       'RealVideo.RealVideo(tm) ActiveX Control (32-bit)',
       'RealPlayer',
```

```
            'SWCtl.SWCtl', // ShockWave player
            'WMPlayer.OCX', // Windows media player
            'AgControl.AgControl', // Silverlight
            'Skype.Detection'];

          // starting to detect plugins in IE
          return this.map(names, function(name){
            try{
              new ActiveXObject(name);
              return name;
            } catch(e){
              return null;
            }
          }).join(';');
        } else {
          return ""; // behavior prior version 0.5.0, not breaking backwards
compat.
        }
      },
```

Obzirom da je način pristupanja dodacima različit u Internet Explorer-u i ostalim pregledačima, metod za preuzimanje liste dodataka sastoji se iz poziva dva pomoćna metoda `getRegularPluginsString()` i `getIEPluginsString()`. U slučaju Internet Explorer-a ne postoji objekat ili osobina objekta na nivou pregledača kojom se iz JavaScript-a može dobiti lista dodataka. Zato polazimo od predefinisanog niza poznatih dodataka, odnosno njihovih oznaka, i pokušavamo da kreiramo `ActiveXObject` objekat i u slučaju da je kreiranje uspešno znači da takav dodatak postoji u pregledaču koji ispitujemo. Za ostale pregledače jednostavno pristupamo promenljivoj `navigator.plugins` i vraćamo nadovezanu listu instaliranih dodataka, spojenu zarezima.

`Canvas` identifikator:

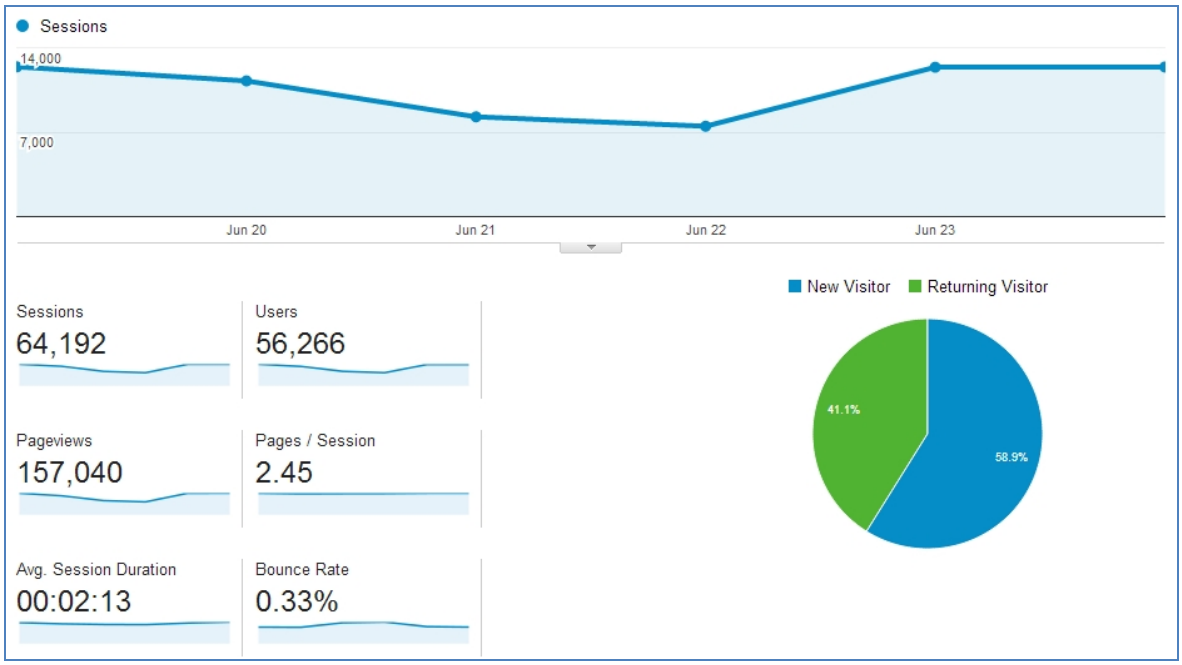
```
    getCanvasFingerprint: function () {
      var canvas = document.createElement('canvas');
      var ctx = canvas.getContext('2d');
      // https://www.browserleaks.com/canvas#how-does-it-work
      var txt = 'portalsrbija <canvas> !@#$^* fp';
      ctx.textBaseline = "top";
      ctx.font = "14px 'Arial'";
      ctx.textBaseline = "alphabetic";
      ctx.fillStyle = "#f60";
      ctx.fillRect(125,1,62,20);
      ctx.fillStyle = "#069";
      ctx.fillText(txt, 2, 15);
      ctx.fillStyle = "rgba(102, 204, 0, 0.7)";
      ctx.fillText(txt, 4, 17);
      return canvas.toDataURL();
    }
```

Prethodni metod sličan je postupku opisanom u 5.5 – počinjemo sa kreiranjem 2d konteksta, a zatim definišemo proizvoljan tekst, podešavamo izgled, boju i druge osobine

teksta i na kraju ispisujemo tekst na `canvas`. Nakon završenog ispisivanja uzimamo Base64 enkodiran string PNG verzije regiona `canvas`-a. U osnovnoj metodi, koja vraća glavni identifikator, nad dobijenim Base64 stringom primenjujemo Sha1 heš koji uzimamo za komponentu glavnog identifikatora.

7. Analiza efikasnosti rešenja

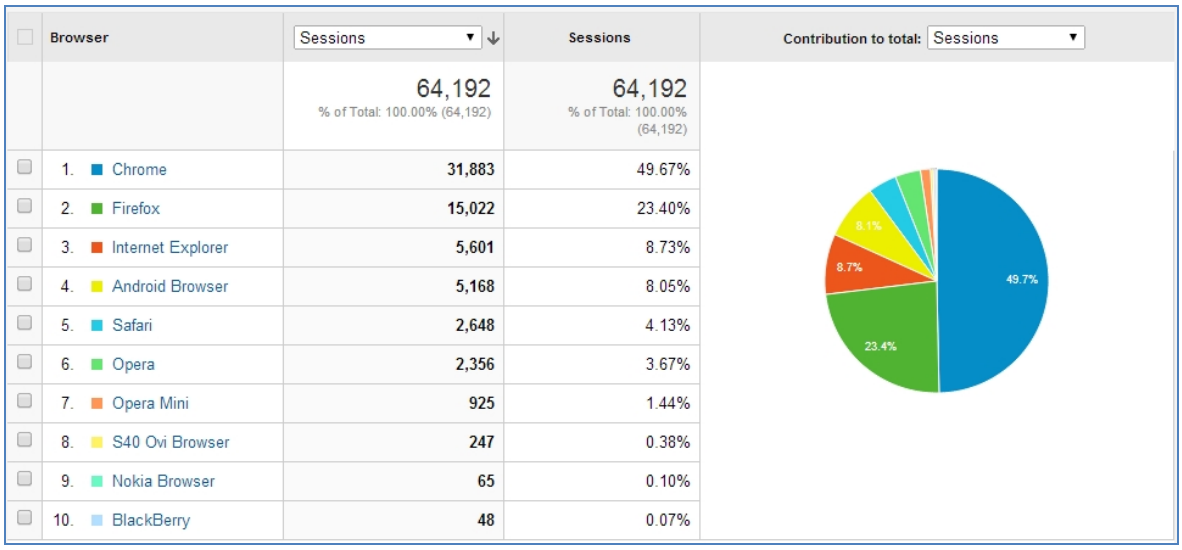
Pre nego što predstavimo rezultate našeg rešenja iznosimo nekoliko uvodnih napomena. Naš uzorak pregledača baziran je na oko 54.000 (prema internim statistikama, 56.266 prema Google Analytics servisu, slika 7) jedinstvenih poseta sajtu www.portal-srbija.com u periodu od 19. do 25. juna 2014.



Slika 7 Statistika posećenosti www.portal-srbija.com, 19. - 26. jun 2014.

Prema Google Analytics terminologiji *jedinstvena poseta (users)* bazira se na globalno jedinstvenom identifikatoru koji server dodeljuje pregledaču prilikom prve posete, sa instrukcijom da ovaj identifikator bude snimljen u kolačić sa vremenskim trajanjem od jedne godine. *Sesija (sessions)* je kontinuriana aktivnost korisnika bez prekida dužih od 30 minuta. Broj *pregledanih strana (pageviews)* je broj klikova na regularne linkove od strane korisnika. *Procenat odbijanja (bounce rate)* je procenat svih korisnika koji su posetili tačno jednu stranu. Ukoliko je definisano dodatno vremenski zasnovano praćenje događaja procenat odbijanja se smanjuje, tako da ga uvek treba uzeti sa rezervom.

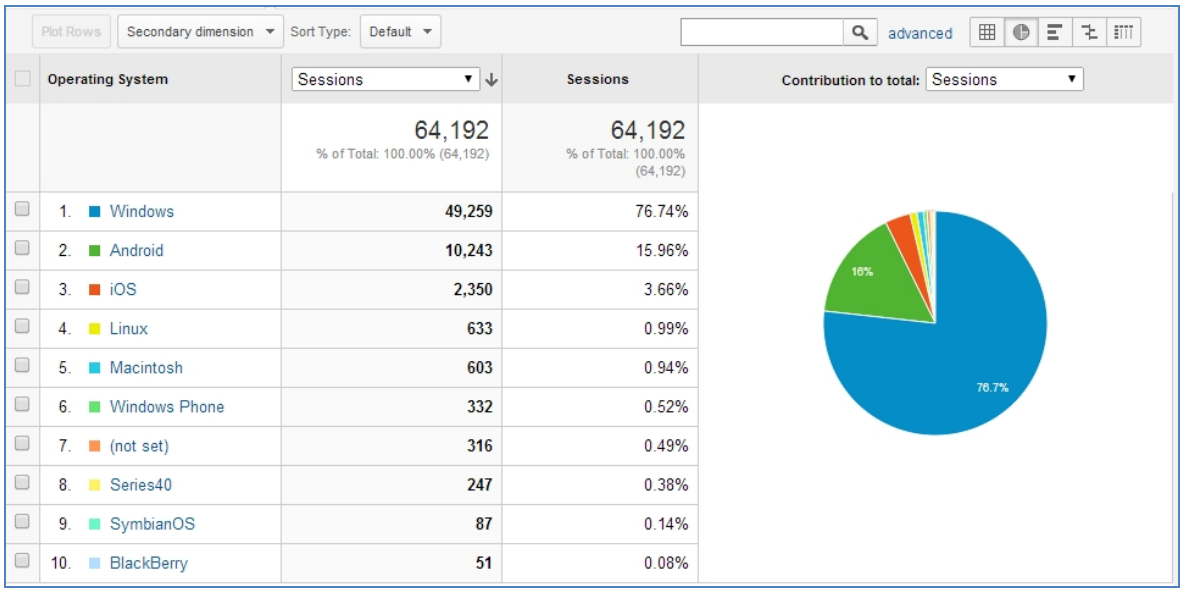
Struktura pregledača prikazana je na slici 8. Kao što vidimo prikazani su i mobilni i desktop pregledači. Sa stanovišta efikasnosti rešenja velika većina (vidi 5.1) pregledača podržava tehnologije koje smo koristili. Ipak detaljnijom analizom utvrdili smo da neka od merenja treba posmatrati posebno za mobilne i desktop pregledače.



Browser	Sessions	Sessions	Contribution to total: Sessions
	64,192 % of Total: 100.00% (64,192)	64,192 % of Total: 100.00% (64,192)	
1. Chrome	31,883	49.67%	
2. Firefox	15,022	23.40%	
3. Internet Explorer	5,601	8.73%	
4. Android Browser	5,168	8.05%	
5. Safari	2,648	4.13%	
6. Opera	2,356	3.67%	
7. Opera Mini	925	1.44%	
8. S40 Ovi Browser	247	0.38%	
9. Nokia Browser	65	0.10%	
10. BlackBerry	48	0.07%	

Slika 8 Struktura pregledača na posmatranom uzorku

Kao što vidimo na slici 9 mobilni operativni sistemi učestvuju sa oko 20% u odnosu na ceo posmatrani uzorak.

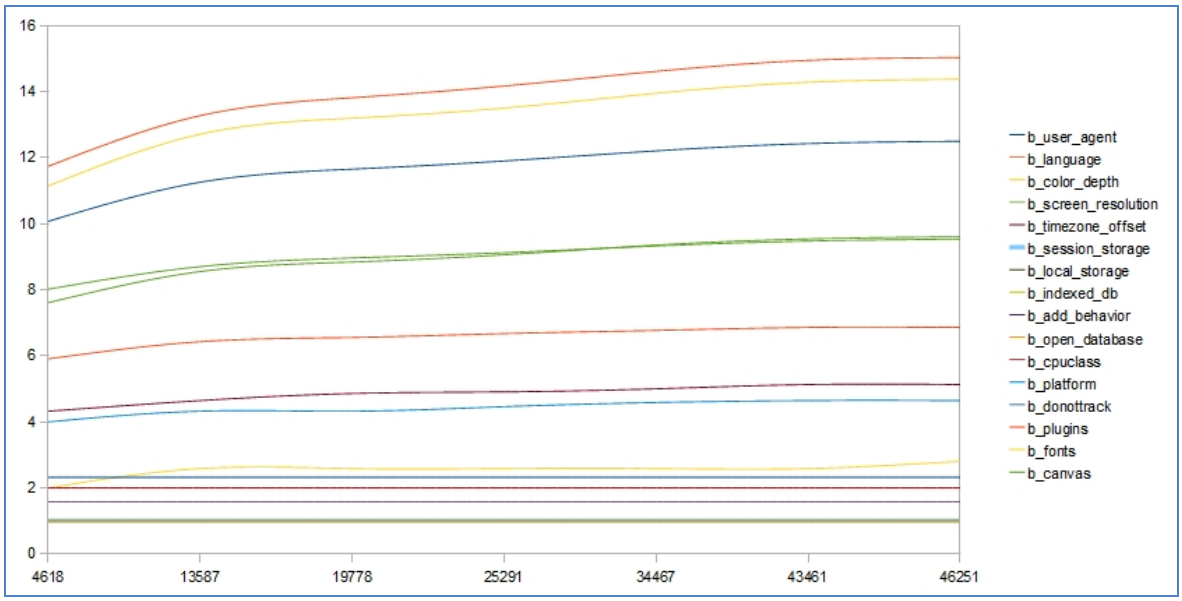


Plot Rows | Secondary dimension | Sort Type: Default | advanced

Operating System	Sessions	Sessions	Contribution to total: Sessions
	64,192 % of Total: 100.00% (64,192)	64,192 % of Total: 100.00% (64,192)	
1. Windows	49,259	76.74%	
2. Android	10,243	15.96%	
3. iOS	2,350	3.66%	
4. Linux	633	0.99%	
5. Macintosh	603	0.94%	
6. Windows Phone	332	0.52%	
7. (not set)	316	0.49%	
8. Series40	247	0.38%	
9. SymbianOS	87	0.14%	
10. BlackBerry	51	0.08%	

Slika 9 Struktura operativnih sistema na posmatranom uzorku

Kao referentno istraživanje uzimamo Panopticlick projekat, obzirom na sličnu metodologiju formiranja identifikatora. Ipak treba napomenuti da je naš uzorak posmatranih pregledača dosta manji, kao i da postoje razlike u parametrima koji su komponente identifikatora. Sa druge strane za Panopticlick projekat mobilni pregledači nisu imali značaj dok su veoma značajni u našem projektu.



Grafikon 1 Promena Šenonove entropije komponenata identifikatora sa rastom broja posmatranih pregledača

Sumarni pregled rezultata implementacije identifikacije klijenata (IDK):

	Binarna entropija			Procenat jedinstvenih ID-ova		
	ukupno	desktop	mobilni	ukupno	desktop	mobilni
IDK	15,49	15,35	12,11	85,78	94,81	44,37
Panopticlick	18,10	-	-	92,40	-	-

Tabela 2 Sumarni pregled rezultata i poređenje IDK i Panopticlick

Kao što se može videti u tabeli 2, binarna entropija (ili sopstvena informacija) u našem eksperimentu je značajno manja, ali vrlo dobra u odnosu na veličinu uzorka. Ako pogledamo procente jedinstvenih identifikatora naš način identifikacije je nešto bolji od Panopticlick-a, odnosno uspeli smo da jedinstveno identifikujemo 94.81% pregledača (smatramo da možemo da poredimo naš rezultat za desktop sa Panopticlick ukupnim rezultatom). Poređenjem binarne entropije komponenata (tabela 3) vidimo da komponente sa najvećom raznovrsnošću imaju bliske vrednosti, bez obzira na veličinu uzorka. To ukazuje da su ove vrednosti tokom vremena relativno stabilne i ograničene, što znači da je za identifikovanje velikog broja pregledača (desetine, ili stotine miliona) potrebno upotrebiti dodatne komponente sa potencijalno većom sopstvenom informacijom.

Komponenta	Binarna entropija	
	IDK	Panopticlick
`plugins`	15,03	15,40
`fonts`	14,37	13,90
`user_agent`	11,74	10,00
`canvas`	9,04	-
`screen_resolution`	8,85	-
`language`	6,55	-
`http_accept`	-	6,09
`video`	-	4,83

timezone_offset	5,00	3,40
platform	4,25	-
color_depth	2,81	-
donottrack	2,32	-
supercookies	-	2,12
cpuclass	2,00	-
add_behavior	1,58	-
indexed_db	1,00	-
local_storage	1,00	-
open_database	1,00	-
session_storage	1,00	-
cookies enabled	-	0,35

Tabela 3 Poređenje binarne entropije IDK i Panopticlick

Razlog za relativno slab rezultat identifikovanja mobilnih pregledača je neprilagođenost algoritma za tu svrhu, kao i činjenice da su mobilne platforme relativno nove i nedovoljno istražene sa stanovišta identifikovanja (tabela 4)

Komponenta	Binarna entropija	
	desktop	mobilni
plugins	15,03	5,52
fonts	14,37	2,00
user_agent	11,74	11,20
canvas	9,04	7,92
screen_resolution	8,85	8,43
language	6,55	6,43
timezone_offset	5,00	4,58
platform	4,25	3,58
color_depth	2,81	2,00
donottrack	2,32	2,32
cpuclass	2,00	0,00
add_behavior	1,58	0,00
indexed_db	1,00	1,00
local_storage	1,00	1,00
open_database	1,00	1,00
session_storage	1,00	1,00

Tabela 4 Poređenje binarne entropije za desktop i mobilne platforme

Kao što se vidi u tabeli 4, komponente sa najvećom binarnom entropijom (a samim tim i sa najvećim uticajem na identfikator) u desktop identifikaciji imaju dosta slabije vrednosti u identifikaciji na mobilnim uređajima. Razlog za ovo je u nemogućnosti dobijanja ili nepostojanju određe komponente. Tako vidimo da komponente `cpuclass` i `add_behavior` uopšte ne postoje na mobilnim uređajima, dok za druge (`fonts`, `plugins`) možemo pretpostaviti da se potpuno drugačije dobijaju od sistema (ili je pristup tim komponentama blokiran za veb aplikacije). Većina komponenata dostupnih kroz

HTML5 API (`canvas, local_storage, session_storage`) imaju slične ili iste vrednosti iz razloga što pregledači na mobilnim uređajima uglavnom podržavaju HTML5.

8. Zaključak

U ovom radu opisali smo, implementirali i praktično testirali jedan algoritam identifikacije klijenata u veb aplikacijama. Algoritam se pokazao kao efikasan, a u nekim slučajevima i efikasniji od referentnih rešenja. Kako smo naveli u sekciji 5 postoje različiti načini za dalje unapređivanje ovog algoritma i, uopšteno, postoji prostor za dalje istraživanje u ovoj oblasti, pre svega u identifikaciji na mobilnim uređajima i boljoj primeni HTML5 funkcija.

Identifikacija pregledača je moćna tehnika sa potencijalno širokom primenom u praksi. Većina Internet kompanija koristi neki oblik identifikovanja pregledača. Iako nije moguće pouzdano utvrditi u koje svrhe se identifikovanje koristi praksa pokazuje da većina kompanija ima odgovarajuća objašnjenja na svojim sajtovima ukoliko koristi neku od tehnika identifikacije uređaja. Suprotno tome, kompanije koje se ne bave identifikovanjem ali koriste usluge prethodno pomenutih kompanija vrlo retko otvoreno naglašavaju da posredno identifikuju korisnike (ovo je naročito slučaj u distribuciji reklama). Sam proces identifikacije je relativno kompleksan i potrebna su određena stručna znanja da bi se razumelo o čemu se tačno radi, što dodatno doprinosi ignorisanju ove tehnike od strane krajnjih korisnika.

Primene identifikacije pregledača uključuju detekciju prevara, zaštitu od neovlašćenog preuzimanja naloga, zaštitu od veb robota i automatizovanog pristupa servisima, menadžment sigurnosti u velikim kompanijama, zaštitu od DdoS (Distibuted denial-of-service) napada, ciljani marketing u realnom vremenu, objedinjavanje sesije na više uređaja i ograničavanje pristupa servisima. Većina kompanija koje se bave identifikacijom specijalizuje se ili za "detekciju prevara i veb sigurnost" ili za "primene u marketingu".

Pitanje privatnosti korisnika i zaštite informacija o ličnosti, odnosno mogućeg povezivanja informacija o uređaju sa infomacijama o ličnosti, još uvek nije dobro definisano ni zakonski uređeno. Postoje podeljena mišljenja da li prikupljanje informacija o korisnikovom uređaju ugrožava privatnost korisnika. Svakako da primena identifikacije korisnika u svrhe istraživanja auditorijuma i ponašanja korisnika na Internetu sama po sebi nema opasnosti po privatnost jer se uglavnom radi o anonimizovanim podacima i sumarnim izveštajima. Međutim, logično "unapređenje" istraživanja auditorijuma je tzv. "personalizacija" sadržaja koja omogućava da se korisniku prijavljenom na servis prikazuju sadržaji (reklame) shodno njegovim interesovanjima. Problem je što postoje osnovane sumnje da se identifikacija uređaja koristi u kombinaciji sa drugim faktorima u svrhu praćenja korisnika i u slučaju kada je odjavljen, tj. kada ne želi da se lično identifikuje. Nadamo se da će istraživanja slična našem doprineti boljem razumevanju i upotrebi tehnika identifikacije.

9. Literatura

[1] P. Eckersley. How Unique Is Your Web Browser? [online] https://panopticlick.eff.org/browser-uniqueness.pdf

[2] RFC [online] http://tools.ietf.org/html/rfc6265

[3] N. Zakas. HTTP cookies explained [online] http://www.nczonline.net/blog/2009/05/05/http-cookies-explained/

[4] S. Kamkar. Evercookie [online] http://samy.pl/evercookie

[5] J. Mena. *Machine Learning Forensics for Law Enforcement, Security, and Intelligence*. Boca Raton, FL: CRC Press (Taylor & Francis Group). ISBN 9-781-4398-6069-4.

[6] U. Luc. Cookieless cookies [online] http://lucb1e.com/rp/cookielesscookies

[7] N. Zakas. How many users have JavaScript disabled? [online] https://developer.yahoo.com/blogs/ydn/many-users-javascript-disabled-14121.html

[8] J. R. Mayer. „Any person ... a pamphleteer“. Senior Thesis, Stanford University, 2009.

[9] N. Nikiforakis, et al. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 541-555, 2013.

[10] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham. Fingerprinting information in JavaScript implementations. In H. Wang, editor, *Proceedings of W2SP 2011*. IEEE Computer Society, May 2011.

[11] T. F. Yen, et al. Host fingerprinting and tracking on the web: privacy and security implications. In *Proceddings of the 19th Annual Network and Distributed System Security Symposium* (NDSS), 2012.

[12] T. Kohno, A. Broido, and K. Claffy. Remote physical device fingerprinting. *Dependable and Secure Computing, IEEE Transactions on*, 2(2):93-108, 2005.

[13] G. Acar et al., FPDetective: Dusting the Web for Fingerprinters, CCS '13 *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, Pages 1129-1140, ACM New York, NY, USA, 2013

[14] V. Vasilyev. fingerprintJS [online] https://github.com/Valve/fingerprintjs

[15] A. Rantakari. Getting a List of Installed Fonts with Flash and Javascript [online] http://hasseg.org/blog/post/526/getting-a-list-of-installed-fonts-with-flash-and-javascript/