Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Híradástechnikai Tanszék

Boda Károly

BÖNGÉSZŐFÜGGETLEN RENDSZERUJJLENYOMAT

MINT WEBES NYOMKÖVETÉSI MÓDSZER KIDOLGOZÁSA ÉS VIZSGÁLATA

KONZULENS

Gulyás Gábor György

BUDAPEST, 2012

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Boda Károly**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2012. 05. 20.

.....

Boda Károly

Köszönetnyilvánítás

Szeretnék köszönetet mondani Gulyás Gábor Györgynek, aki rugalmas hozzáállással, kötelességén túl segített a kutatásban és a diplomaterv elkészítésében. Továbbá szeretném megköszönni Földes Ádám Mátének, hogy szakmai észrevételeivel és ötleteivel segítette munkámat a cikkírásban, és segített a Böngészőfüggetlen ujjlenyomat projekt 2.0 weboldal tartalmának létrehozásában is. Köszönetet mondok dr. Székely Ivánnak, aki hozzájárult a nemzetközi tesztelő csoport kialakításához, illetve a sajtóesemény színvonalas lebonyolításához.

Rövidítésjegyzék

Rövidítés	Feloldás
AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
CSS	Cascading Style Sheets
DNT	Do Not Track
DOM	Document Object Model
HTTP	Hypertext Transfer Protocol
ID	Identifier
IP	Internet Protocol
IS	Isolated Storage
ISP	Internet Service Provider
JS	JavaScript
JSON	JavaScript Object Notation
LSO	Local Shared Object
MAC	Media Access Control
NAT	Network Address Translation
PET	Privacy Enhancing Technology
PIE	Persistent Identification Element
TCP	Transmission Control Protocol
TOR	The Onion Router
UAS	User Agent String
UID	User Identifier
URL	Uniform Resource Locator
OS	Operation System

Alkalmazások rövidítései

Rövidítés	Feloldás
CH	Chrome (Google) böngészőprogram
FF	Firefox (Mozilla) böngészőprogram
FG	FireGloves kiterjesztés
IE	Internet Explorer (Microsoft) böngészőprogram
MS	Microsoft
OP	Opera böngészőprogram
SA	Safari (Apple) böngészőprogram

1 Bevezető

Az információs és kommunikációs technológiák (röviden IKT) napjainkban meghatározó szerepet töltenek be. Ennek alapvető eleme az internet, legkiemelkedőbb szereppel pedig a web böngészéshez kapcsolódó, azaz a hypertext alapú kommunikáció rendelkezik. Mára jelentős üzleti iparággá nőtte ki magát a webes reklámozás és kereskedelem, gondoljunk csak a weboldalakon éktelenkedő színes, hangos reklámokra. Ennek közvetlen vonzata az erre a célra szánt média hatékonyságának mérése, és a felhasználók szokásainak megismerése, profitnövelési célból.

A web böngészés során nagy mennyiségű – az átlagos felhasználó elől rejtett – adatot továbbítanak a böngésző programok, ezen felül még többet lehet kinyerni a különféle webes technológiákon keresztül. Ezeket implicit adatoknak nevezzük, mert nem a felhasználó adja meg őket (szemben a szándékosan feltöltött explicit adatokkal), hanem a webes szolgáltatások megrendelői, fejlesztői és üzemeltetői gyűjthetik össze „megfigyeléssel”. Többféle célból történhet az adatok gyűjtése, mint például a tranzakció, munkaviszony azonosítása, látogató számlálás, statisztikakészítés, látogatók profilírozása, és kereskedelem hatékonyságának növelése [1].

Fontos szerepet kap tehát a felhasználók azonosítása akkor is, hogyha nincsenek bejelentkezve egy oldalra sem, csupán „egyszerű látogatók”. Ugyanis ez esetben megfigyelt tevékenységükből lehet megismerni profilírozás által a személyek érdeklődési körét, ízlésvilágát, vagy például vallási, politikai orientációit [2]. Az efféle azonosításra azonban nincs egységesen meghatározott módszer, a fejlesztőknek a rendelkezésre álló webes eszközök alkalmazására kell, hogy hagyatkozzanak. Azonban az internetes privátszféra tudatosság folyamatosan fejlődik, emellett az eszközök hatékonysága is változik, ezért időről-időre meg kell újulniuk az azonosítási technikáknak is [2].

Számos módszer létezik azonosításra, pl. IP cím alapú és tárolásra épülő módszerek, mint például a sütikben tárolt azonosítók, amelyet egy adott oldal meglátogatása esetén a szolgáltatók könnyen kiolvashatnak az oldalba épített „detektorok” (vagyis távolról lehívott képek) segítségével [1]. Problémát jelent viszont az IP cím változása vagy megosztása (pl. NAT esetén), és tárolás esetén pedig az, hogy a felhasználó törölheti a gépéről az eltárolt azonosítót, valamint további probléma, hogy

jogi szabályozás alá kerültek az EU-ban a süti alapú technikák [3]. Ezen problémák összessége miatt az utóbbi években intenzív kutatás alá kerültek az ún. ujjlenyomat technikák, amelyet az EU szabályozása miatt már a piaci szereplők is alkalmaznak[1]. Ezen módszerek lényege, hogy a böngésző program API-ján keresztül, esetleg különböző kiegészítők (plugin) felhasználásával kinyerik a kliens eszköz konfigurációs adatait, amelyből egy egyedi azonosítót készítenek. Ezt nem szükséges eltárolni, mert a kliens visszatérésekor ismét el tudják készíteni, amelynek előzményeit kikereshetik az adatbázisból.

Az ujjlenyomat technikák jelenleg is aktívan kutatás alatt állnak, az első meghatározó kutatás a Panopticlick projekt volt[2], melynek célja a böngészők ujjlenyomatának egyediségének vizsgálata volt. Ennek megfelelően böngésző-ujjlenyomatot készít, az eljárása tárolásmentes, gyors, és pontos, amennyiben a kliensen Flash vagy Java plugin elérhető. Ez utóbbiak segítségével a böngészőben elérhető adatok mellett a rendszerre telepített betűtípusok listáját is lekérdezi, majd az adatok összessége alapján vizsgálja a kliens egyediségét, az adatbázisban lévő többi ujjlenyomathoz képest.

Ahogy korábban utaltam rá, az ujjlenyomat módszerek már jelen vannak a webes hirdetési piacon: egy BlueCava nevű cég ezen a technikán alapuló nyomkövetést kínál a hirdetőknek, szolgáltatóknak. Állításuk szerint módszerük megbízhatósága 99%-os, és működik a legkülönfélébb eszközökön is (pl. set-top-box). Vizsgálataim alapján kiderítettem, hogy módszerük tárolással kombinált, ugyanis elsőként a perzisztens tárolókban keresi az azonosítót, és ha ott nem áll rendelkezésre, akkor elkészíti az ujjlenyomatot. Hatékonyságának vizsgálatára teszteket készítettem, amikből kiderült, hogy valóban egy „agresszív módszerről" van szó, nehéz megszabadulni a létrehozott azonosítótól – de nem lehetetlen, ahogy ezt az általam készített védelmi alkalmazásról szóló fejezetben be is mutatom.

[1] BlueCava Press release on a "cookie-less" device identification technique in the area: http://www.bluecava.com/news-release/bluecava-releases-cookie-less-device-identification-technology-for-online-advertisers/

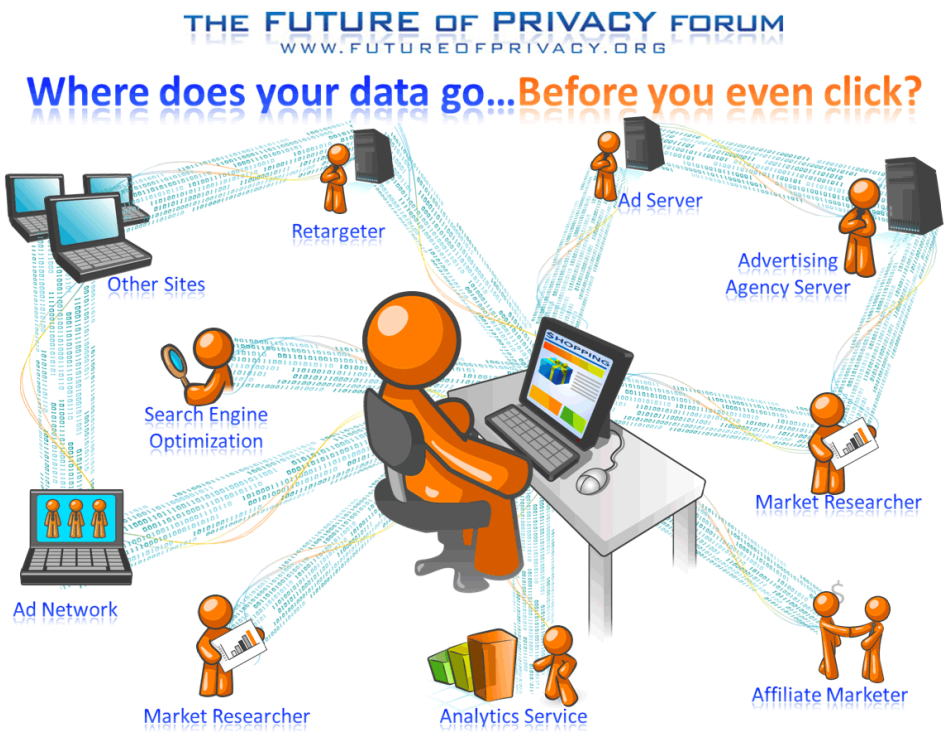
[2] http://panopticlick.eff.org

2 A webes nyomkövetés és ujjlenyomat módszerek

Ebben a fejezetben bemutatom a probléma szereplőit, az ismert és alkalmazott nyomkövetési módszerek működését, illetve a létező és jövőben elképzelhető védekezési megoldásokat.

Kik férnek hozzá az adatokhoz?

A böngészés alapvetően háromszereplős: felhasználók, internetszolgáltatók és a webkiszolgáló szerverek elegendőek a működéséhez. Ennek ellenére manapság nagyon ritka, hogy csak ezek vannak jelen. Általában legalább egy külső megfigyelő szolgáltató részt vesz a folyamatban, de nagyon elterjedt a különféle reklámszolgáltatások integrálása is (ld. 1. ábra).



1. ábra - A webes világ szereplői

Gyakorlatilag két fő csoportba oszthatók a szereplők. Egyik oldal a felhasználók csoportja, akik privát szférájuk megsértése nélkül szeretnék használni a webet, másik oldal a felhasználókat megfigyelők csoportja, akik a felhasználók érdekeivel szemben

álló tevékenységet folytatnak – megfigyelik őket, információt gyűjtenek róluk, befolyásolják őket, vagy korlátozzák a szolgáltatást. Másképp fogalmazva, a felhasználók célja az adatok védelme, szemben a kereskedelmi tevékenységet folytató, illetve cenzorként jelenlévő szereplőkkel. Tevékenységük alapján több szerepkörbe sorolhatjuk a résztvevőket [4].

1. *Felhasználók*: Általában böngészésre, levelezésre, internetes vásárlásra, ügyintézésre, közösségi szolgáltatások igénybevételére használják az internetet.
2. *Internetszolgáltató (ISP)*: A fizikai kapcsolatért felelős, minden egyes csomagról tudomása van. Naplózza a felhasználók adatforgalmát, tiltásokat is alkalmazhat, együttműködhet más szolgáltatókkal (napló fájlok egyeztetése). Egyértelműen azonosítja a felhasználót.
3. *Cenzúrázó szervek*: Szabályozzák az elérhető tartalmat, megfigyelik a felhasználók tevékenységét.
4. *Hirdetők*: Profil alapú célzott hirdetéseket jelenítenek meg a felhasználók által megtekintett weboldalakon (általában harmadik félként), beágyazott formában. Képesek listát építeni a felhasználók által látogatott weboldalakból, ismerhetik a felhasználók internetezési szokásait.
5. *Webes üzletek*: Profil alapú dinamikus árazásra képesek.
6. *Adatgyűjtők*: Profilírozzák a felhasználókat, és adat kereskedelemmel foglalkoznak. A profilokat eladják más szolgáltatóknak, kereskedőknek. Ide tartoznak az önkéntes adatszolgáltatáson alapuló szolgáltatások (pl. közösségi portálok), kereső szolgáltatások, forgalomanalízis és piackutatási szolgáltatások, és az információs szuperhatalmak (többnyire ingyenes) szolgáltatásai is.
7. *Kutatók, elemzők*: A szolgáltatók átadhatják adatbázisukat elemzésre, vagy egyéb kutatási célból (általában anonimizált formában). Ide tartoznak a speciális automatizált mechanizmusok, pl. predikciós algoritmusok, amelyek segítik a cégvezetők döntéseit.

A veszélyforrások három nagy kategóriába sorolhatók, attól függően, hogy milyen adatokat tekintünk:

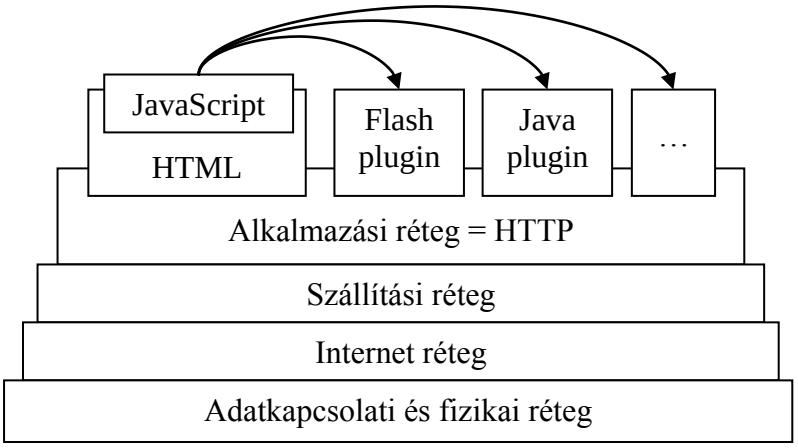
1. „Információs szuperhatalmak”, nagy szolgáltatók szolgáltatásai (pl. Google ingyenes szolgáltatásai): implicit és explicit adatokhoz férnek hozzá

2. Profilírozás publikus adatforrások, önkéntes adatszolgáltatás alapján (pl. keresőkben, közösségi hálózatokban megadott adatok): csak explicit adatokhoz való hozzáférés
3. Nyomkövetéses profilírozás: implicit adatokhoz való hozzáférés

A nyomkövetéses profilírozás célja a profilépítés, ami lehet pszeudonim (szám jellegű azonosítóval ellátott), és azonosított (a felhasználó személyére utaló, vagy azt pontosan megjelölő). Megvalósítása a felhasználó tevékenységének rekonstrukciójával történik, vagy a felhasználónál keletkezett naplók, adatbázisok alapján.

Mit lát a szolgáltató?

A böngésző program a TCP/IP modellen alapuló HTTP protokollt használja a kommunikációhoz, a szolgáltató tehát ezt teljes egészében látja (ld. 2. ábra).



2. ábra - A TCP/IP modell kiegészítve az alkalmazási réteg komponenseivel

A modellt felülről lefelé haladva mutatom be. Legfelső rétege az alkalmazási réteg, az IP alapú nyomkövetést kiváltó technikák mind ezt használják az azonosító létrehozásához szükséges adatok továbbítására. A böngésző program automatikusan csatol HTTP fejléceket, sütiket, és ha van egyéb információ, az is ezen a szinten kerül átadásra – akár a normál működés, akár nyomkövetés részeként.

A második réteg a szállítási réteg, amit a modell nevében is szereplő TCP (Transmission Control Protocol) valósít meg. Ez biztosítja a HTTP számára a sorrendhelyes, hibamentes átvitelt. Nem sok azonosításra alkalmas adatot nyerhető belőle, csupán a kliens kommunikációs portja, ami viszont nem fix, egy bizonyos tartományon belül elég gyakran változik.

A harmadik réteg az internetréteg, amelyben IP (Internet Protocol) működik, a végpontok közötti kommunikációért felelős. Kiolvasható belőle a tranzakciót indító kliens számítógép címe (IP cím), ami bizonyos környezetben már önmagában elég jó azonosítóként szolgál. Ha az ISP dinamikus IP címet használ, lehetőség van ennek módosítására meghatározott tartományon belül, pl. újracsatlakozással juthatunk másik címhez. Azonban fix, vagy ritkán változó IP cím esetén annak lecserélése nem triviális feladat, ezért ilyen közegben kiváló azonosítóként szolgál. Természetesen megoldható proxy kiszolgálón vagy anonimizáló hálózaton (pl. TOR) keresztüli csatlakozással, de ez az egyszerű felhasználók képességeit meghaladja.

A negyedik és ötödik, legalsó rétegek az adatkapcsolati és fizikai réteg. A hálózati eszközök (csomópontok) egymás közötti fizikai szintű adatátvitelét valósítja meg, valós fizikai címekkel kommunikál. Ebbe a szolgáltató már nem lát bele, aminek következtében a kliens gép hálózati eszközének fizikai címét (MAC address) nem tudhatja meg (egyébként az globális azonosítóként működne). Azonban a lehallgatás elleni védelme fontos lehet webes szolgáltatások szempontjából, amint azt a FireSheep alkotói is megmutatták[3], hiszen lehallgatással a munkaviszonyok ellophatóvá válnak.

Alapvető technikák: IP, tárolás alapú nyomkövetés és kiegészítők

A nyomkövetési technikák a webes technológiákkal párhuzamosan fejlődnek, ellenben a webes privátszféra védelme mindig pár lépéssel hátrébb jár [2]. A fejezetben ezeknek a módszereknek a működéséről (és elemzéséről) számolok be a technikai oldalról, a kezdetleges, ma már nem használható módszerektől kezdve egészen a jelenleg is alkalmazott, aktív kutatási területként nyilvántartott módszerekig.

2.1.1 IP alapú nyomkövetés

Ahhoz, hogy a szerver elküldje a kliens gépnek a lekért tartalmat, ismernie kell a kliens hálózati címét (IP cím). Ebből triviálisan következik az IP alapú felhasználó azonosítási módszer, amely egyben a legrégebbi is. Az Internet térhódításának köszönhetően lassan az összes lehetséges 32 bites IPv4 cím kiosztásra kerül, a szolgáltatók többsége ezért áttért a dinamikus IP cím használatára és a hálózati címfordítás alkalmazására (NAT), kibővítve ezzel a szűkössé vált tartományt (bár ez

[3] http://codebutler.com/firesheep

utóbbi az otthoni és a céges hálózatok miatt is gyakori). Emiatt ellehetetlenedett az IP alapú azonosítás módszere, hiszen a dinamikus IP cím mindig más eszközt azonosít, a hálózati címfordítás miatt pedig nem tudjuk, hogy az adott IP cím egy NAT képes eszközt vagy egy végpontot azonosít, ezen felül a címfordítás többrétegű is lehet. A mobil hálózatok kizárólag dinamikus IP címet használnak, a módszer tehát ez esetben nem használható.

Az előbbiek igazak általánosságban az IPv4-re, hiszen még kevés helyen álltak át az IPv6 használatára. Habár manapság már egyre kevesebb a fix cím, esetenként jól alkalmazható a különböző böngészőkből (vagy akár eszközökről) érkező azonosítók (felhasználók) párosítására. Az IPv6 nyomkövetés szempontjából sokkal hatékonyabb lesz, ugyanis a nagy címtartománynak köszönhetően minden cím konkrét eszközt azonosíthat (és tartalmazza a hálózati eszköz fizikai címét), ezáltal egyféle globális azonosítóként is használható lesz. A szabványban [5] definiáltak adatvédelmi kiegészítést, ami hogyha engedélyezve van, akkor az operációs rendszer véletlenszerű fizikai címet csatol a kapott hálózati prefixhez, és így egy nagyon rövid életű IPv6 címet generál. Privát szféra szempontjából ez kedvező, mivel a megfigyelhető viszonyok hossza lerövidül, és az IP alapú nyomkövetés ismét használhatatlanná válik. A modern operációs rendszerekben alapértelmezésként engedélyezve van ez a funkció.

2.1.2 Süti alapú nyomkövetés

A sütik kisméretű, kulcs-érték párokat tartalmazó szöveges fájlok a kliens számítógépén (pontosabban böngésző programjában), amik minden tranzakciónál oda-vissza utaznak a kliens és a szerver között a HTTP protokoll fejlécében. Eredetileg az állapot bevezetése volt a céljuk az állapotmentes HTTP protokollba [4], amely lehet az azonos klienstől érkező lekérések összekapcsolása, de akár a különböző futásokban tárolt eredmények tárolása is (pl. bevásárló kosár tartalma). (Ennek megfelelően a legelterjedtebb alkalmazása az ún. „session cookie”, azaz munkamenet azonosító süti, amellyel megvalósítható pl. a felhasználó beléptetése egy rendszerbe.) Felépítésük összetettebb a kulcs-érték párnál, általában rendelkeznek egy lejárati időponttal is (expiration time), illetve egy doménnel és elérési útvonallal a doménen belül (csak ezen a részen elérhető a süti). Tartalmuk elérése és módosítása történhet a szerver és a kliens oldalon futtatott programból egyaránt, plusz a böngészőprogramok interfészt nyújtanak a felhasználónak azok menedzselésére (megtekintés, törlés).

Ezt a funkciót használják ki a süti alapú nyomkövetési technikák (third party cookie). Működésük alapja, hogy a külső – harmadik féltől származó – objektumok (pl. kép) lekérésekor a HTTP fejlécben a kéréssel együtt elküldésre kerülnek az oldalon beállított sütik, illetve az URL referer (ld. 2.1.5 fejezet) is – az objektum „környezete". A szerver oldalon ezt naplózzák, amelyből rekonstruálható a felhasználó által megtekintett weboldalak listája. Ilyenek lehetnek például a bannerek, beágyazott hirdetések, és a közösségi portálok megosztó gombjai.

A módszer a legtöbb felhasználó esetében a mai napig jól működik, mert háttérismeret hiányában vagy hanyagságból nem törődnek a sütik rendszeres törlésével, harmadik féltől származó sütik tiltásával. Nem működik viszont, hogyha a felhasználó odafigyel a biztonságára, rendszeresen törli a sütiket, vagy újabban privát böngészési módot alkalmaz. Ráadásul soha le nem járó sütik is létrehozhatóak, amik további problémát jelenthetnek.

2.1.2.1 Hirdetések, külső szolgáltatók képei

A harmadik féltől származó sütik tiltására a legtöbb böngészőprogramban van lehetőség, azonban ennek van egy megkerülési módja: az objektumok JavaScripttel való beszúrásakor az URL-hez sütiből kiolvasott azonosítót lehet csatolni, a külső szolgáltató tehát ugyanúgy hozzájuthat, mint sütik használatával. Ezt követően ugyanúgy naplózhatja az URL referert, és rekonstruálhatja a meglátogatott oldalak listáját.

2.1.2.2 Web poloska

Kisméretű, 1x1 pixeles kép (leggyakrabban átlátszó GIF formátumú kép), amit például weboldalaknál a látogatottság megfigyelésére, vagy e-mailekben az üzenet megtekintésének detektálására használnak. Weboldalak esetén általában harmadik féltől származik, és megjelenítésekor sütit is beállítanak, így lehetőség van a későbbi azonosításra [6]. A módszer elég elterjedt [7], új formáját a közösségi hálózatok megosztó/kedvelő gombjainak – és más widgeteinek – integrálása jelenti, hiszen ezek a web poloskához hasonlóan harmadik féltől származó objektumok, ugyanúgy alkalmazhatók nyomkövetésre [8].

2.1.3 Flash sütik (LSO)

A Flash technológia által lehetőség nyílt a böngészőfüggetlen tárolásra is, mivel valamennyi a kliensen futó böngésző alkalmazás ugyanazt a plugint használja, és a

Flash is rendelkezik sütikkel, az ún. LSO-kal (Local Shared Object), amelyek a sütikhez hasonlóan működnek [9]. Az így beállított Flash sütik elérhetőek valamennyi böngészőből (az adott doménen), így arra is lehet használni, hogy egy böngészőben beállított sütit egy másik böngészőbe átmásoljunk a szerver közreműködése nélkül (ld. Evercookie). A nyomkövetésbeli alkalmazásuk esetén az ún. „tracking-cookie”-hoz hasonlóan ezt is eltérően nevezik, az ilyen Flash sütik az ún. PIE-ok [10] (Persistend Identification Element). A Microsoft Silverlight technológiája is rendelkezik hasonlóan működő IS (Isolated Storage) tárolóval, így hasonló célokra az is használható.

Azonban a böngésző csak JavaScripten keresztül hívhatja le (miután a Flash alkalmazás betöltődött), mivel eltérően a sütiktől csak a Flash-en belül érhető el, és nem utazik a HTTP fejlécen belül együtt a lekérésekkel. Azonban a követés ugyanúgy megvalósítható, a lekérdezett értéket vissza lehet küldeni a szerverre AJAX kérésen keresztül, további felhasználása a törölt böngésző sütik visszaállítására is (ezt „cookie respawn”-nak vagy „zombie cookie”-nak nevezzük).

Egy 2009-es amerikai felmérés szerint a 100 legnépszerűbb oldalból 89-en alkalmazták az azonosítási eljárást, sőt, kormányzati oldalakon is [10]. Sokáig csak körülményes módon – kézzel vagy külső programmal – lehetett törölni a Flash sütiket, az Adobe 2010-ben jelentette be újítását: a privát böngészési mód közben létrehozott Flash sütik törlődnek a privát mód elhagyásakor. A modern böngészőprogramok (2011 májusa óta) a sütik hagyományos módon való törlésekor küldenek egy-egy jelet a tárolóval rendelkező kiegészítőknek, hogy töröljék az adott doménhez tartozó értékeket; ez a Flash 10.3 verziójában megvalósításra[4] került.

2.1.4 Evercookie

A tárolás alapú nyomkövetés elleni védekezés nehézségét demonstrálandó, Samy Kamkar létrehozott egy törölhetetlen, ön-visszaállító megoldást, ami a dedikált süti tárolón kívül, több más tárban is elhelyezi az azonosítót, és ha az törlődött valamelyik tárolóból, egy másikból visszaállítják azt [11]. Ilyen tárolók az LSO, IS, HTTP ETag[5], gyorsítótárazott (PNG formátumú) képek[6], a böngésző `window.name` attribútuma[7], userData Storage[8], valamint a HTML5-tel megjelent Session Storage,

[4] http://blogs.adobe.com/asset/2011/05/advancing-flash-player-privacy-and-security.html
[5] Enity Tag: azonosító alapú módszer a web cache hatékonyságának növelésére.
[6] Azonosító tárolása RGB képpontokban, kiolvasás canvas segítségével.
[7] Perzisztens JavaScript objektum tároló, domén független, különböző oldalakon kiolvasható.
[8] IE által alkalmazott tárolási módszer

Local Storage, Global Storage és Database Storage. Egy időben működött az előzményekben elhelyezett (History Stealing alapú) azonosító tárolás is, azonban a modern böngészőkben erre nincs lehetőség. Az Evercookie technológia majdnem az összes ismert tárolóban elhelyezi a megadott azonosítót, és ha törlődött valamelyikből, többségi szavazás alapján visszaállítja azt.

2.1.5 Az URL-refererhez kapcsolódó problémák

Az URL-referer annak az oldalnak az URL címe, amelyikről indítottuk a lekérést az aktuális oldalra (pl. linkre kattintást, vagy kép lekérést). A HTTP fejléc hordozza, a felhasználó alapvetően nem tudja befolyásolni. Egy weboldal szolgáltató használhatja például statisztika készítésére, hogy mely címekről érkezett a legtöbb látogató, vagy kereső oldalról érkező átkattintásnál a kulcsszavak megfigyelésére, illetve kép beágyazásánál a kép környezetének érzékelésére.

2.1.6 Meta-információk automatikus felfedése

Az RSS és Atom hírcsatornák, valamint a kezdőlap – vagy a legutóbb megnyitott lapok – automatikus letöltése lehetőséget nyújt a szolgáltatóknak a felhasználók internetes szokásainak felfedésére (pl. napi első letöltés). Az IP címet is automatikusan felfedi, hiszen süti vagy más módszer használatával össze tudja kapcsolni csatlakozási helyünk címeit (pl. otthon, munkahely, kávézó), sőt, ezek földrajzi elhelyezkedését is ismerheti (Geolocation).

Danis Mihály diplomatervében [12] beszámol róla, hogy a hagyományos nyomkövetési technikák RSS-ben is működnek. Még akkor is, hogyha tilt valamit a felhasználó, jó eséllyel megkerülhető valamilyen módszerrel, pl. JavaScript tiltása egyszerűen egy `<iframe>` (beágyazott keret) használatával megkerülhető. A vizsgált 40 csatornából 11-ben alkalmazták a nyomkövető módszerek valamelyikét, volt, ahol Flash sütiket is.

2.1.7 Azonosításon alapuló módszerek

A tárolás alapú módszerekkel ellentétben nem egy eltárolt azonosító kiolvasása a cél, hanem a böngésző program speciális tulajdonságainak felhasználásával egy azonosító létrehozása. Ennek eljuttatása a szerverre többnyire AJAX POST metódussal történik, vagy egyszerűen a változót eltárolják valamelyik tárban. A módszer alapvetően akkor jó, hogyha egymástól független látogatások alkalmával ugyanazt az azonosítót

generálja a rendszer (IP címtől, doméntől, tárolók ürítésétől függetlenül). Kombinálni természetesen lehet – és érdemes is – más eljárásokkal a hibatűrés javításának érdekében.

2.1.7.1 A History Stealing módszer: már a múlté

A böngészési előzmények célja, hogy a felhasználó számára követhetővé tegyék mely oldalon járt már vagy sem, és érzékeny volta miatt ez az előzmény lista adatvédelmi okokból nem kérdezhető le explicit módon, azonban a History Stealing technika a védelmet megkerülve képes volt erre. A meglátogatott hivatkozásokat a böngészőprogram eltérő színnel jelöli, hogy jelezze a felhasználó felé, hogy már járt az oldalon (CSS `visited` pszeudo osztály), és ezt használta a History Stealing módszer. JavaScript segítségével egy hivatkozás listát (láthatatlanul) beszúr a weboldalba, majd lekérdezi az egyes linkek színét. Hogyha eltért valamelyik az alapértelmezettől, akkor a felhasználó már járt az adott címen, egyébként még nem [13].

Amellett, hogy a módszer alkalmas azonosításra, remek lehetőséget kínál a felhasználó érdeklődési körének feltérképezésére, és ennek ismeretében célzott hirdetések elhelyezésére, illetve dinamikus árazásnál is hatékony lehet: például egy web bolt detektálhatja, hogy egy konkurens üzlet oldalán egy terméket megtekintett a látogató, és annak szándékosan alacsonyabb árat adhat. Ennél súlyosabb probléma, hogy közösségi portálokon is alkalmasnak bizonyult valódi személy azonosítására, az egyedi URL címmel rendelkező csoporttagságok ellenőrzésével [14]. Ez vezetett végül a technikát támogató JavaScript hiányosság befoltozásához [15]. (Továbbra is használható a látogatott oldalak megkülönböztető színnel történő jelölése, viszont a DOM-ban nem jelenik meg az eltérés, ezzel lehetetlenné téve annak JavaScript-es detektálását.)

2.1.7.2 A Panopticlick projekt

A mai passzív – tárolást nem igénylő – technikák kutatásának alapjául szolgáló Panopticlick projekt egészen újszerű megközelítésben vizsgálja a nyomkövetést: tárolt azonosító helyett böngésző adatainak ujjlenyomatozhatóságát, illetve a begyűjtött adatbáziban a böngésző egyediségét vizsgálja. Felhasználja a böngésző által küldött HTTP fejléceket (User Agent String, HTTP Accept fejlécek), és a böngészőből elérhető állandó jellegű adatokat (kiegészítők listája, időzóna beállítás, képernyő beállítás, telepített betűtípusok listája, süti engedélyezés állapota, valamint az elérhető tárolók

állapota). A telepített betűtípusok listájának lekérdezésére két módszert alkalmaz, egy Flash, és egy Java alapút, amennyiben az előbbi nem áll rendelkezésre. Ez utóbbiak segítségével gyors és pontos böngésző azonosítást valósít meg, a kutatási beszámoló [16] szerint egymillió látogatónak a 83,6%-a egyedi ujjlenyomattal rendelkezik, de Flash vagy Java plugin használatával (telepített betűtípusok detektálása) az érték 94,2%-ra nő. Hátránya, hogy böngésző- és pluginfüggő, hiszen támaszkodik a User Agent Stringre, a plugin listára, valamint legalább egy plugin bekapcsolt állapotára.

A projekt demonstrációs oldalán bárki elvégezheti a tesztet[9]. Futtatás után megjeleníti, hogy a felhasználó hány tesztelő közül bizonyult egyedinek adatai alapján. Alatta egy táblázatban listázza a begyűjtött adatokat, bináris entrópia értékét, valamint adatonként az egyediségi halmaz méretét. A projektről készült publikációból kiderül [16], hogy a legnagyobb entrópia értékekkel a plugin lista (15,4), a telepített betűtípusok listája (13,9), és a User Agent String (10,0) rendelkezik. A weboldalon védelmi ajánlás[10] is olvasható, melyben szerepel többek között a JavaScript kikapcsolása, TorButton használata, illetve javaslattétel a privát böngészési mód továbbfejlesztésére. Ezek azonban nem elegendők: a TorButton használata nem gátolja meg az ujjlenyomatozást, a JavaScript kikapcsolása pedig elveszi a modern weboldalak funkcionalitását (ld. 0 fejezet).

2.1.7.3 A böngésző- és eszköz fingerprinting történeti áttekintése

A 2011-ben hatályba lépett EU-s „cookie direktíva” [3] szerint viselkedésalapú reklámot csak a felhasználók előzetes hozzájárulásával lehet alkalmazni. A webes szolgáltatások (pl. webáruház) alapvető funkcionalitásához szükséges sütiken kívül (pl. munkamenet azonosító) bármi más célt szolgáló süti alapú adattárolás (és hozzáférés) előtt tájékoztatni kell a felhasználót az adatkezelés céljáról. A szabályozás így megnehezíti a profilírozást, ami miatt az ezt alkalmazó résztvevők alternatív módszereket keresnek, amilyen például a süti-mentes nyomkövetési technikák. A módszerek további előnye, hogy nem érzékenyek a további perzisztens tárolók ürítésére sem, és ahogy jelen dolgozatból kiderül, megvalósítható böngésző-független módon is.

[9] https://panopticlick.eff.org
[10] https://panopticlick.eff.org/self-defense.php

A Panopticlick projekt után a következő nagyobb kísérlet a Nemzetközi PET Portál és Blog által indított Rendszerujjlenyomat projekt volt[11], melynek célja böngészőtől és kiegészítőktől független ujjlenyomat[12] előállítása volt. A gyűjtött adatok kielemzésében már részt vettem, amelynek eredményeit nemzetközi konferencia publikációban foglaltuk össze [17]. A kísérlet során majdnem 1000 ujjlenyomatot sikerült összegyűjteni, amely alapján sikerült igazolnunk, hogy a plugin-ek nélkül detektált betűkészletek listája is elégséges információforrás lehet a követéshez, azonban a böngészőfüggetlen ujjlenyomat koncepciójához további adatgyűjtésre volt szükség.

Ezen okok miatt, illetve más, javított módszerrel elindítottuk a projekt folytatását, a *Böngészőfüggetlen ujjlenyomat teszt 2.0*-t[13], amely tesztoldal fejlesztését én végeztem. Az ujjlenyomat részeként több újdonság is szerepel, mint pl. a plugin lista mentése, finomított betűkészlet detektálás, evercookie alapú azonosító tárolás (az ujjlenyomatok követéséhez), valamint fő különbség, hogy kifejezetten bátorítjuk a kísérletet elvégzőket, hogy különböző böngészőkben is lekérdezzék az ujjlenyomatukat. Az ujjlenyomat kísérlet indítását kellő médiaérdeklődés kísérte, számos médium is beszámolt róla[14].

A második kísérletünk előkészítésével párhuzamosan a piacon is megjelent egy szereplő, a BlueCava nevű amerikai cég, aki a mi ujjlenyomat koncepciónkéhoz hasonlóval reklámozza magát, és kifejezetten az EU süti szabályozás által korlátozott hirdetőket célozza meg.

A bécsi egyetemen is kutatják a témát: kidolgoztak egy módszert, amely a JavaScript motorok eltérő implementációját használja ki, az ECMAScript[15] specifikációnak való megfelelést ellenőrzi egy tesztsorozattal[16], amely több mint ötezer apró tesztet tartalmaz. Működését 189 felhasználóval tesztelték, az algoritmus átlagosan 90 ms alatt futott le, és minden felhasználót sikeresen azonosított hiba nélkül. Emellett több nem akadémiai körökből származó kísérlet is folyik, ezek hivatkozásától „szürke” jellegük miatt eltekintek.

[11] Rendszerujjlenyomat: http://pet-portal.eu/fingerprint/
[12] Böngésző specifikus jellemzők elhagyásával, és egy újfajta kiegészítő-mentes betűtípus detektálási módszerrel.
[13] Böngészőfüggetlen ujjlenyomat teszt 2.0: http://fingerprint.pet-portal.eu
[14] Sajtómegjelenések az eseményről: http://pet-portal.eu/press/#ujjlenyomat2
[15] http://test262.ecmascript.org/
[16] http://en.wikipedia.org/wiki/Sputnik_(JavaScript_conformance_test)

Az ujjlenyomat módszerek technikai háttere

A böngészőprogram lekéréseknél számos fejlécet[17] küld a felhasználó elől rejtett módon, aki egyszerű eszközökkel nem képes beavatkozni ebbe. Ezek közül a relevánsakat az 1. táblázat tartalmazza névvel, leírással, és egy-egy példával[18] szemléltetve.

Adat neve	Leírás	Példa
Accept	elfogadható média típusok	`text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`
Accept charset[19]	elfogadható karakterkódolások	`ISO-8859-2,utf-8;q=0.7,*;q=0.3`
Accept encoding	elfogadható adattömörítő eljárások	`gzip,deflate,sdch`
Accept language	elfogadható válasz nyelvek	`hu-HU,hu;q=0.8,en-US;q=0.6,en;q=0.4`
Referer	annak az oldalnak a címe, amelyről megnyitottuk a linket	`http://google.com`
User agent	a böngészőprogram és a rendszer bizonyos tulajdonságait leíró karakterlánc	`Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.162 Safari/535.19`
DNT[20]	Do Not Track bit, a felhasználó követés tiltásának állapotát jelzi	`1`

1. táblázat - Releváns adatot hordozó HTTP fejlécek

A böngésző JavaScript API-ján keresztül kinyerhető adatok önmagukban nem egyediek, viszont kombinálva ezeket, bizonyos környezetben már azonosítóként használhatók, hasonlóan ahhoz, ahogy egy személyt sem azonosít külön a lakóhelyének irányítószáma, életkora és neme, de ezek együttesen már igen [18]. Ehhez hozzávéve a telepített betűtípusok és kiegészítők detektálásával kapott listát egy egyedi azonosítót lehet létrehozni. A 2. táblázat összefoglalja a releváns adatok elérését, leírását, és egy-egy példát[21] megjelenésükre (a teljes táblázat böngésző támogatási megjegyzésekkel a mellékletben látható).

[17] HTTP fejléc definíciók: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html
[18] A példák Google Chrome (18.0) böngészővel készültek.
[19] Csak Chrome böngészők által használt fejléc.
[20] A Google 2012 év végéig tett ígéretet a DNT fejléc beépítésére.
[21] A példák Mozilla Firefox (11.0) böngészővel készültek, Flash plugin installálva volt.

Adat elérése	Leírás	Példa
navigator.userAgent	User Agent String (böngészőleíró)	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:11.0) Gecko/20100101 Firefox/11.0
navigator.platform	A böngésző fordítási platformja	Win32
navigator.language	Nyelvi beállítás kódja	hu-HU
navigator.plugins	A telepített kiegészítők listája. Tartalmazza a kapcsolódó média típusokat (számmal elnevezett objektumokként) és a lista hosszát, ezen túl a plugin leírását, a futtatott fájl nevét és részletes verzióját, valamint a plugin nevét.	A lista egy eleme: [...] length: 2 description: Shockwave Flash 11.2 r202 filename: NPSWF32_11_2_202_228.dll version: 11.2.202.228 name: Shockwave Flash item: [native code] namedItem: [native code]

navigator.mimeTypes	A böngésző által értelmezett média típusok listája. Tartalmaz egy leírást, a kapcsolódó kiegészítő objektumot (enabledPlugin), a hozzá tartozó fájlnév kiterjesztéseket (suffixes), valamint a média típusát.	A lista egy eleme:
		description:
		Adobe Flash movie
		enabledPlugin: {
		length: 2
		description:
		Shockwave Flash 11.2r202
		filename:
		NPSWF32_11_2_202_228.dll
		version: 11.2.202.228
		[...]
		}
		[...]

navigator.doNotTrack	Nyomkövetés elleni kérelem.	unspecified
navigator .geolocation	GeoLocation alapú helymeghatározó szolgáltatás. Általában a felhasználó engedélye szükséges hozzá.	getCurrentPosition()
navigator .cookieEnabled	Cookie engedélyezés állapota.	true
screen, window.screen	A monitor beállításainak tényleges, illetve a böngésző számára elérhető méreteit, és a színmélység értékét tartalmazza. Kiszámítható belőle a tálca mérete.	height: 900 width: 1600 pixelDepth: 24 colorDepth: 24 availWidth: 1600 availHeight: 860
Date .getTimezoneOffset	Időzóna eltérés a rendszer beállítása alapján.	-120

2. táblázat - JavaScript API-n keresztül elérhető releváns adatok[22]

A jelenlegi ujjlenyomatmódszerek fontos eleme a telepített betűtípusok detektálása, bár vannak módszerek, amelyek a teljesítmény mérésén alapulnak. A Flash és Java alapú megoldások azonos módon működnek, és azok is alkalmazhatóak plusz

[22] Információforrások: https://anonymous-proxy-servers.net/wiki/index.php/JonDoFox_sources

információk detektálására[23]. Először egy JavaScript kód beszúrja őket az oldalba kis 1x1 pixeles objektumként (vagy már eleve a HTML kódban vannak), így nem észrevehetőek, majd miután az alkalmazások programja lefutott, a rendszertől lekérdezi, és elérhetővé teszi az eredményt JavaScripten keresztül. Az oldal betöltésének készenléti pillanatában a beszúrt objektumok még nem állnak készen[24], ezért a betűtípuslista lekérését késleltetni kell.

Létezik azonban egy másfajta, pluginoktól független betűtípus lekérdezési eljárás is, amelyet a Rendszerujjlenyomat és a Böngészőfüggetlen ujjlenyomat teszt 2.0 is alkalmaz. A módszer HTML DOM elemek `offsetWidth` és `offsetHeight` attribútumán alapul, azt használja ki, hogy a beszúrt tartalom méretei (szélessége és magassága) változnak a kiválasztott betűtípustól függően. Azonban szükség van egy előre definiált betűtípus listára, amit végigjárva ellenőrizhető az egyes betűtípusok megléte azáltal, hogy ugyanazon szöveg méreteit összevetjük különböző betűtípusok esetén az alapértelmezett betűtípushoz képest. Az összehasonlításhoz szükséges kiindulási alap a „sans” betűcsalád, ennek oka, hogy sok böngészőben ez az alapértelmezett. Ha egy betűtípus nincs installálva a gépre, akkor a beszúrt elem mérete megegyezik az alapértelmezett betűtípussal beszúrt elemével, egyébként eltérő. A módszer tehát nem a ténylegesen telepített betűtípusokat adja meg, csak a listában szereplőket ellenőrzi – ellenben a plugin alapú detektálással.

Védekezési módszerek

A nyomkövetéssel szemben a legkevesebb, amit tenni lehet, a böngésző biztonsági beállításainak helyes konfigurálása, például harmadik féltől származó sütik elfogadását le kell tiltani, célszerű a sütik kilépéskor történő automatikus törlését is bekapcsolni. Modern böngészőkben hasonlóképpen beállítható a cache ürítése, és az előzmények törlése a programból való kilépéssel. A nem használt kiegészítőket érdemes letiltani (vagy törölni), ezzel csökkenthető a plugin lista mérete.

A helyes beállításokon túl privát böngészési mód használata javasolt, kimondottan a tárolásos módszerek ellen hatékony – az Evercookie módszert teljes egészében képes hatástalanítani. Használatakor a sütik, HTML5 local storage és a

[23] Erre demonstrációs oldal például: http://ip-check.info/?lang=en

[24] Időbe telik az objektumok létrehozása, hiszen le kell tölteni tartalmukat, külön alkalmazást (plugin) kell indítani hozzá, és inicializálni kell. Ez az idő milliszekundumos nagyságrendben mérhető, de mindenképp számolni kell vele.

gyorsítótár ugyanúgy funkcionál, mint normál módban, azonban tartalmuk csak a memóriában lesz jelen – nem készül másolat belőlük a háttértárolón – és a privát mód elhagyásakor törlődnek. Ennek okán, privát böngészés után normál módba váltva nem lesznek elérhetők a privát módban létrehozott tartalmak [19]. Fordított esetben (normál mód után privát mód) is hasonló a helyzet, kivétel ez alól a Safari böngésző, amelyben a módváltást követően elérhetők maradnak a sütik és a HTML5 local storage.

Hatékonyabb védekezés valósítható meg kiegészítők alkalmazásával, amelyekkel szűrni lehet a JavaScript kódokat, illetve blokkolás Flash és Java objektumok esetén. A példákon kívül számos másik kiegészítő is létezik, és némelyikben beállítható, hogy csak a megtekintett weboldaléval azonos doménről töltse le és futtassa a JavaSript kódot (ún. whitelisting). Az objektumok automatikus blokkolásakor egyenként lehet kiválasztani a megtekinteni kívánt médiumokat – ennek eredményeképp az 1x1 pixeles nyomkövetésre vagy detektálásra használt alkalmazások kudarcot vallanak.

Azonban nem elegendő, ha a szükséges lépéseket csak az alkalmazási rétegben tesszük meg, szükséges a látogató hálózati anonimitásának biztosítása is. IP cím elrejtése történhet ún. anonim proxy szerveren keresztül, vagy MIX-ek segítségével [20], mint amilyen például a TOR elnevezésű anonimizáló hálózat [21]. A valódi IP cím ezek mellett is könnyen felderíthető, például egy olyan Flash vagy Java alkalmazás beágyazásával, ami közvetlen kommunikál a szerverrel, kap egy session ID-t, és azzal indít egy GET kérést a szerver felé ahol pedig megtörténik az összepárosítás.

Mindezen funkciók együttesének szolgálására megszülettek az anonim böngészők [22]. Ebből több típus is létezik[25] [23]. A kliens alapúak általában hálózati anonimizáló szolgáltatással együtt működnek, azonban mivel ezek fejlesztői nem az alkalmazásért, hanem a hozzá kapcsolódó infrastruktúraüzemeltetésért kérnek ellenszolgáltatást, ezért az ingyenes verzióik általában túlterheltek, lassúak. Az online verziók hátránya, hogy egyrészt nagyon lerombolják[26] a felhasználói élményt, másrészt meg kell bízni a harmadik félben, aki továbbra is látja az összes elérhető adatot, és immáron a teljes webes tevékenységet is.

Az említett védekezési megoldások egyike sem véd jelenleg az ujjlenyomat alapú nyomkövetés ellen. A létező módszereket a 6. fejezetben fejezetben ismertetem.

[25] http://pet-portal.eu/articles/view/28/?set_language=eng
[26] A különböző tartalomszűrők sok esetben helytelenül működnek, letiltják a rendeltetésszerű működéshez szükséges kódrészeket is.

3 Modern ujjlenyomatmódszerek kritériumai

Az ujjlenyomat kísérletek célja ennek a speciális sérülékenységnek a demonstrálása, méghozzá olyan módszer által, amely független a használt böngészőprogramtól és pluginoktól, nem érzékeny az IP cím változására, és tűri a tárolók és gyorsítótár törlését (tehát elkerülve a felhasználók pillanatnyilag elérhető védekezési lehetőségeit). Lényeges szempont továbbá, hogy működése észrevétlen legyen, azaz ne foglalja le látványosan a böngészőprogramot, és ne terhelje minden lekéréskor nagy mennyiségű adattal a hálózatot. Jelen dolgozat tárgya az információ alapú ujjlenyomatmódszerek, ezért a többire (például teljesítmény alapú és cache alapú módszerek) nem térek ki.

Azonosító komponensek vizsgálata

A potenciális információforrásokat három szempont alapján vizsgáltam meg: az adatok *elérhetősége*, *élettartama*, és *módosíthatósága* szempontjából. Az elérhetőség mezőben „✓” jelzi, ha az öt legnépszerűbb böngésző mindegyikéből elérhető az adat, „✗” jelzi, hogyha csak részlegesen érhető el, vagy speciális feltételhez kötötten (például felhasználói engedély), ezekben az esetekben univerzálisan nem elérhető. Az élettartam a környezeti változások adatokra való hatását fejezi ki (pl. böngésző szoftver frissítése, új plugin telepítése). Módosíthatóság jellemzi, hogy a felhasználó képes-e közvetlenül (a böngészőprogramon belül, szakmai tudás, ill. kiegészítők használata nélkül) az érték megváltoztatására amellett, hogy a felhasználói élmény nem csorbul (például a képernyőfelbontás vagy az időzóna eltolódás módosítható a rendszerbeállításoknál, de ez körülményes, és a felhasználói élmény rovására megy). A 3. táblázat tartalmazza az egyes adatok tulajdonságait, a plusz információt nem hordozó adatok (például a User Agent string részei külön) nem kerültek bele.

Géptermekben – ahol csupa egyező konfigurációval rendelkező gép van – az ujjlenyomatmódszer önmagában elméletileg is kevés, hiszen nem érhető el olyan adat pluginok bevonása nélkül, ami alapján megkülönböztethetők lennének (pl. hálózati kártya MAC címe). Ha erre az esetre is szeretnénk felkészíteni a rendszert, célszerű perzisztens tárolást is alkalmazni, mint az evercookie módszer.

Adat	Elérhetőség	Élettartam	Módosítható
Komm. port	✓	Általában oldal lekérésenként, vagy újratöltésenként változik	×
IP cím	✓	Változó[27]	×[28]
Accept fejlécek	✓[29]	Cserénél, frissítéskor változhat	×
URL Referrer	✓	Oldal lekérésenként változhat	×
User Agent string	✓	Böngésző váltásnál, frissítéskor változhat[30]	×
DNT	✓[31]	Nem változik	✓
Fordítási platform[32]	✓	Nem változik	×
Nyelvi beállítás	✓	Nem változik	×
Plugin lista	✓[33]	Plugin telepítésekor, tiltásakor változik	✓[34]
Média típusok listája	×[35]	Plugin telepítésekor, tiltásakor változik	✓
Processzor típusa	×[36]	Nem változik	×
Build ID	×	Cserénél, frissítéskor változik	×
GeoLocation	×[37]	Helyzetfüggő	×
Cookiek engedélyezése	✓	Nem változik	✓
Képernyő adatok	✓	Nem változik	×
Időzóna eltérés	✓	Nem változik	×
Időzítés adatok	✓	Oldal lekérésenként változik, teljesítmény és hálózat függő	×

3. táblázat - Alkalmazási rétegből elérhető adatok tulajdonságai

Hatékony azonosító létrehozása

Abban az esetben, ha az azonosító nem tartalmaz böngésző- és plugin specifikus jellemzőket, azaz csak a rendszerre vonatkozó beállítások szerepelnek benne, akkor rendszerujjlenyomatként is funkcionál. Az így létrehozott azonosító robusztus, hiszen

[27] Lehet fix, dinamikus (ezen belül is változó gyakorisággal frissülhet), proxy, kávézó, mobil, stb.
[28] Dinamikus IP cím cserélhető, vagy speciális csatlakozással módosítható (anonimizáló hálózat), de kézzel nem írható át.
[29] Karakterkódolás nélkül, mert az csak Chromeból érhető el.
[30] 2011-ben lerövidült a legtöbb böngésző esetében, régebben tartalmazott plugin verziókat, .NET verziót, eszköztár verziókat, stb.
[31] Chrome jelenleg nem támogatja, kikapcsoltnak tekintjük.
[32] Platformkód, amely a böngésző fordítási módjára utal, például „Win32" = 32 bites Windows.
[33] Internet Explorerből speciális úton lehet kinyerni, és koránt sem olyan részletes, mint a Netscape alapú böngészőknél.
[34] Plugin tiltásával, új plugin telepítésével, a lista lekérdezésének tiltásával módosítható.
[35] Internet Explorer esetén nem érhető el.
[36] Internet Explorer és Firefox esetén elérhető, különböző módszerekkel.
[37] A felhasználó engedélye szükséges hozzá, ami a felhasználó riasztásával jár.

böngészőváltás, vagy plugin telepítés (vagy törlés) nem befolyásolja az értékét, komponenseit ezek figyelembevételével célszerű kiválasztani. A megfelelő azonosító megalkotása a lehetséges komponensek vizsgálatával kezdődik.

3.1.1 IP cím függetlenség

Egyre népszerűbbek a hordozható számítógépek (pl. notebook, tablet pc, okostelefonok), az embereknek magukkal viszik az eszközt munkahelyre, találkozókra, kikapcsolódáshoz, utazáskor, stb. Mivel mindezekben a környezetekben esetenként is akár új IP címet kaphat az eszköz, megfontolandó az IP cím használata, kiváltképp, hogy önmagában azonosítóként tekintsünk rá.

Azonban az azonosító egyediségét növelik akár a kis entrópiájú információforrások is, ezért tervezői döntés, hogy felhasználjuk-e az IP cím egy részét. Az első két bájt általában a szolgáltatót (ISP) határozza meg, ezért dinamikus IP cím cseréje esetén is változatlan marad, ami asztali gépeknél használható lehet. Mindazonáltal a későbbi elemzés céljából (pl. rekordok összeköthetősége nem triviális esetekben) praktikus az IP cím eltárolása. Időpont hozzárendelésével továbbá megkülönböztethetők a csatlakozási helyek, pl. otthon és munkahely.

3.1.2 Böngészőprogram függetlenség

Eléréséhez a 3. táblázatban felsorolt információforrások helyes megválasztása szükséges. Böngésző független rendszerujjlenyomat készítéséhez azokat célszerű felhasználni, amelyek *elérhetők minden böngészőből*, a táblázat *élettartam* oszlopában *nem változik* szerepel, *módosítása* pedig *nem lehetséges*. Ezek például: fordítási platform, nyelvi beállítás, képernyő adatok és időzóna eltérés (ez nem egy zárt lista, szükség esetén bővíthető).

A legmeghatározóbb komponens a JavaScript fontdetektáló algoritmussal létrehozott betűtípusok listája, hiszen pluginok nélkül előállítható, és ez által mindig felhasználható az azonosítóban. Azonban másképp kell kezelni, mint a fenti információforrásokat, hiszen értéke nem csak a telepített betűtípusoktól, hanem a böngészőprogram implementációjától és a tesztelés bemeneteként megadott betűtípus adatbázistól is függ. A különböző böngésző implementációk eltérően használják a betűtípusokat, ezért nincs rá garancia, hogy az egyes fontok megjeleníthetők minden böngészőben – ahol nem jeleníthető meg, ott az alapértelmezett font érvényesül. A feladatot nehezíti, hogy rendszertől függően is lehetnek eltérések, ezért a népszerű

rendszerek mindegyikén tesztelni kell. A betűtípus adatbázist tehát úgy kell összeállítani, hogy azonos gépen – és rendszeren – bármely böngésző esetén ugyanazt az eredményt adja. A tesztelés lokális gépen láthatóan nehéz, hosszan tartó folyamat, emellett az egyes betűtípusok beszerzése sem könnyű, vannak licenccel védettek is.

3.1.3 Plugin függetlenség

A telepített pluginok listája szintén eltérhet böngészőprogramonként, még akkor is, hogyha ugyanazokat a pluginok vannak engedélyezve. Ennek oka szintén a böngésző implementációra vezethető vissza (Chromeban pl. beépített Flash lejátszó van, IE a többitől eltérő Acrobat verziót jelenít meg, stb.). Az IE egyáltalán nem támogatja a `navigator.plugins` kollekciót, a pluginokat csak beszúrásos tesztelés útján lehet lekérdezni. Mindez oda vezet, hogy a pluginlista által hordozott értékes információt nem lehet közvetlenül felhasználni, mert nem érhető el böngészőfüggetlen módon. A betűtípusokhoz hasonlóan meg kell keresni a módját, hogy azonos rendszerben eltérő böngészőkkel ugyanazt az eredményt kapjuk, hiszen a cél böngészőfüggetlen azonosító generálása.

Azonosító komponenseként nem célszerű olyan adatot választani, amely csak plugin használatával kérdezhető le, mert sok múlik a felhasználón – telepítette-e, engedélyezi-e. A betűtípus detektálás megoldható plugin nélkül, ahogy a többi rendszerjellemző is lekérdezhető JS-ből. Mindemellett a pluginokon keresztül begyűjthető adatokat is érdemes lehet eltárolni az ujjlenyomat mellett későbbi elemzés céljából (pl. konkrét betűtípus lista), vagy például robusztusabb, több lábon álló módszer létrehozásához.

Erőforrásigény és hatékonyság

A nyomkövetők szempontjából további fontos szempont, hogy a felhasználó ne fogjon gyanút a lassabb oldalbetöltés miatt, illetve így el is hagyhatja a felhasználó az olyan oldalakat, ahol ilyen követési technikát alkalmaznak.

A sebesség szempontjából kritikus a JS betűtípus detektáláshoz használt adatbázis számossága, méretével mind az adatforgalom mérete, mind a futásidő arányosan nő. Több ezer betűtípus ellenőrzésére tehát nincs lehetőség, a listát az információérték megtartása mellett célszerű minimalizálni. A módszer előnye viszont,

hogy nem kell rá várni, mint a plugin alapú módszerek esetében, az oldal betöltésének idejében elvégezhető.

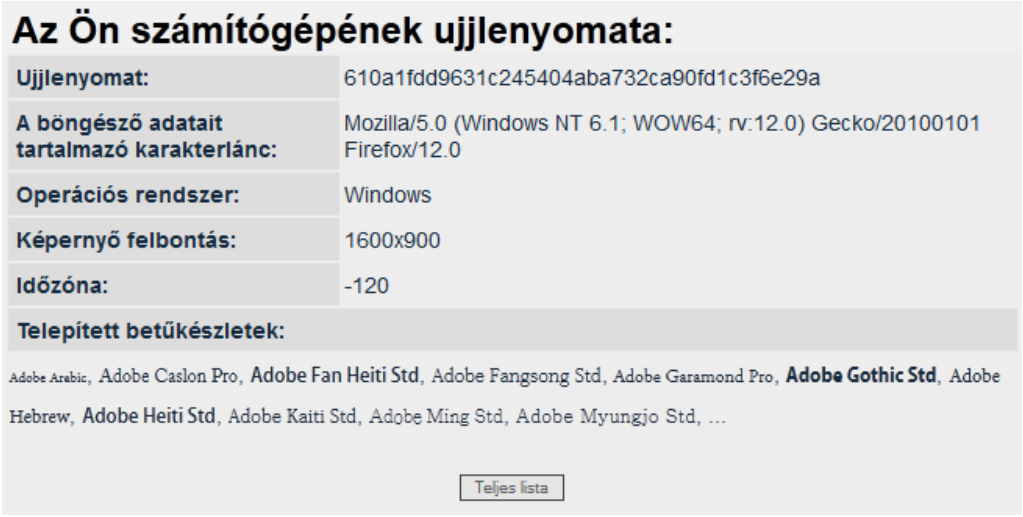
Tovább növelhető a hatékonyság perzisztens tárolási módszer használatával. Eltárolható a létrehozott ujjlenyomat több tárolóban, és csak akkor kell újra kiszámítani, hogyha kitörlődött a tárolókból, vagy a meglévő másolatok alapján már egyértelműen nem állítható helyre. Egy tárolt érték lekérdezése nagyságrendekkel gyorsabb, mint az ujjlenyomat ismételt létrehozása. A kettő kombinációja is növeli az ujjlenyomatozó módszer hatékonyságát, hiszen ha valamely összetevő értékének megváltozása miatt eltérő ujjlenyomat születik, akkor a tárolt érték alapján párosítható a kettő, követhető a változás.

4 Rendszerujjlenyomat teszt

Az első ujjlenyomat tesztet a Nemzetközi PET Portál és Blog[38] indította 2010 szeptemberében. Az ujjlenyomatozó módszert Gulyás Gábor György készítette, munkáját a Panopticlick projekt ihlette. Bekapcsolódásomkor hat hónapja folyt az adatgyűjtés, feladatom a módszer hatékonyságának elemzése, továbbfejlesztése volt. Ehhez az Eötvös Károly Intézet (EKINT) – a gyűjtött adatok adatkezelője – rendelkezésemre bocsátotta az ujjlenyomat adatbázist.

Adatok gyűjtése

Egy webes demonstrációs felületen[38] bárki lekérdezhette a rendszer által generált ujjlenyomatát (bár mérsékelten, de a kísérlet a sajtóban is megjelent[39]). A „Kérem az ujjlenyomatom” gombra kattintva elindul a kliens oldali ujjlenyomat script, összegyűjti az adatokat, majd POST metódussal elküldi a szervernek, ahol összeáll az ujjlenyomat, és tárolásra kerül az adatbázisban (ennek struktúrája látható a 4. táblázatban). A szerver végül visszaküldi a kliens oldalnak a generált azonosítót, amely megjeleníti a felhasználó számára az ujjlenyomatot és a lekérdezett adatokat (3. ábra).



3. ábra - Az ujjlenyomat és a begyűjtött adatok megjelenítése

[38] http://pet-portal.eu/fingerprint/
[39] A sajtómegjelenések itt elérhetőek: http://pet-portal.eu/press

A módszer célja egyedi azonosító létrehozása böngésző-, plugin-, és szolgáltató független módon. Ennek eléréséhez az azonosító a következő összetevőkből épül fel: rendszer tulajdonságok (pl. operációs rendszer, képernyőfelbontás), telepített betűtípusok listája, és az IP cím első két bájtja (4. táblázat). A böngésző- és pluginfüggetlenség elérése érdekében a plugin lista lekérése teljesen kimaradt, a betűtípus detektálás pedig JS programmal történik, egy válogatott – először böngészőfüggetlennek feltételezett – fontokból álló lista alapján. A szolgáltató függetlenség részben teljesül (pl. dinamikus IP címek esetén), de szolgáltató váltásra azonban érzékeny. Ez a továbbfejlesztett böngészőfüggetlen ujjlenyomatban került csak megoldásra.

A 4. táblázat a forrás adatbázis ujjlenyomat táblájának (*fingerprints*) struktúráját ábrázolja. Az azonosító részét képző adatok mellett – kutatás céljából – tárolásra került a *létrehozás időpontja*, a választott *nyelvi beállítás*, a generált *azonosító hash változata*, a *User Agent string* és a *teljes betűtípus detektálás* eredménye. A detektáláshoz használt betűtípusok táblájának (*fingerprint_fonts*) szerkezetét az 5. táblázat mutatja.

Mező	**Leírás**
id	A rekord egyedi azonosítója
locality	A használt nyelvi beállítás (hungarian/international)
user_id	A generált felhasználó azonosító
user_id_short	Az előbbi hash formában
created	Rekord létrehozásának ideje
ip	Látogató IP címe titkosított (hashelt) formában
user_agent	User Agent string
os	Operációs rendszer
screen	Képernyő felbontás
timezone	Időzóna
fonts	Jellemző betűkészletek rövid, „hexa” ábrája
fonts_bin	Betűtípusok listája bináris formában

4. táblázat - A *fingerprints* tábla struktúrája

Mező	**Leírás**
id	A rekord egyedi azonosítója
safe	„Biztonságos-e” a betűtípus
title	Betűtípus neve

5. táblázat - A *fingerprint_fonts* tábla struktúrája

4.1.1 Az adatbázis bővítése

A *fonts_bin* mező a detektált betűtípuslista bináris formája[40]. Hossza megegyezik a betűtípus adatbázis számosságával, bitjei az egyes betűtípusokat név szerint növekvően rendre azonosítják. Egy bit értéke `1`, ha telepítve van az adott pozícióban álló betűtípus, egyébként `0`. Ez az ábrázolás azonban az elemzés során nem ad elég szabadságot a speciális lekérdezések futtatásához, ezért létrehoztam az ujjlenyomatok és betűtípusok kapcsolatát reprezentáló táblát (*relation*). A tábla feltöltése: minden egyes ujjlenyomathoz (rekordhoz) a *fonts_bin* karakterenkénti feldolgozása szükséges, a folyamat automatizálására készítettem egy php scriptet.

A *user_agent_string* több komponensből áll, többek között tartalmazza a böngészőprogram nevét és részletes verziószámát is. E két adat az elemzés során fontos szűrési és csoportosítási feltétel, ezért az ujjlenyomat táblát kibővítettem egy mezővel (*browser_ver*), amiben a böngésző neve (pl. „Firefox") és verziószámának egy tizedesre csonkolt formája (pl. „3.6") szerepel. Az adatok kinyerése reguláris kifejezés illesztéssel lehetséges, ezt is automatizáltam egy php script segítségével. A kutatás közben további mezőkkel is bővítettem az adatbázist, ezeket a kapcsolódó részeknél tárgyalom.

4.1.2 Adatbázis statisztika

A hat hónapos adatgyűjtési folyamat alatt 989 tesztfuttatás volt 615 különböző IP címről, és 662 különböző felhasználó azonosító (UID) jött létre. Ezen adatok arra engednek következtetni, hogy az azonosítók többsége különböző számítógépről lett létrehozva. Ezt megerősíti az is, hogy 439 IP cím és 480 UID csak egy alkalommal fordul elő, illetve 565 esetben egy IP címhez csakis egy UID tartozik. Mindazonáltal bizonyos IP címek több különböző felhasználó azonosító mellett is megjelennek, aminek három oka lehet:

1. a tesztelő különböző rendszerbeállításokkal is futtatta a tesztet,
2. ugyanarról az IP címről különböző eszközökkel kapcsolódtak,
3. eltérő betűtípus detektálás történ különböző böngészőprogramokkal[41].

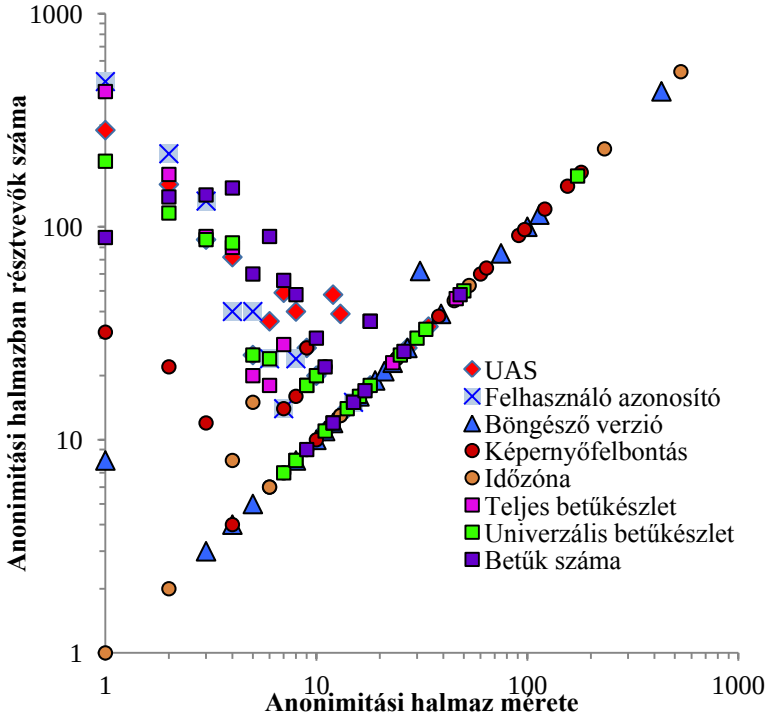
Az adatbázis lehetőséget adott egyéb érdekességek megfigyelésére is. 450 különböző UAS érkezett, ebből 284 pontosan egyszer szerepelt, ami azt jelenti, hogy a böngészők közel 30%-a teljesen egyedi már akkor, ha csak az UAS-et tekintjük. A

[40] Valójában bináris értékekből összefűzött string.
[41] Habár a válogatott betűtípuslista elméletileg böngészőfüggetlen, elképzelhető, hogy bizonyos rendszer-böngésző kombinációban mégsem az, nem volt tesztelve minden lehetséges esetre.

különböző (teljes) betűtípus listák száma 588, ebből 431 csak egyszer fordul elő, nagyon jól azonosítja a böngészőpéldányokat. 649 látogató magyar nyelvű, és 340 idegen nyelvű megtekintés volt, ami tükrözi a hazai és nemzetközi tesztelők arányát, és látszik, hogy a projekt nemzetközi hirdetése kevésbé volt hatékony.

A tesztelők 68 különböző képernyőfelbontást használtak, legnépszerűbb az 1280x800 (18%), és az 1280x1024 (16%). A többség széles képernyőkre (widescreen) jellemző felbontást használt, speciálisan notebookokra jellemzőt. A legnépszerűbb (fonts, screen, timezone) hármas számossága 26, amit 9 különböző IP címről hoztak létre; tehát a legnépszerűbb beállítások kombinációját alkalmazva is csak egy kisméretű anonimitási halmazba lehet bekerülni.

A script 26 különböző böngészőverzióban[42] futott, legkedveltebb a Mozilla Firefox 3.6 volt, a tesztek 44%-ával. A teszt megmutatta, hogy a rendszer- és böngészőjellemzők nagyon változatosak, a 4. ábra ezt szemlélteti, anonimitási halmazok szerinti bontással. (Az ábra rajzolata nagyon hasonlít a Panopticlick projekt azonos tárgyú ábrájához, összehasonlítás a függelékben) Többszöri futtatással a látogatóknak csak 28%-a próbálkozott, a többség mindössze egyszer tesztelt.



4. ábra - A gyűjtött adatok anonimitási eloszlása [17]

[42] Egy tizedesre csonkolt verziószám alapján csoportosítva.

4.1.3 Összehasonlítás a Panopticlick adatbázisával

Mivel az ujjlenyomat adatbázis rekordjainak számossága relatíve kicsi, szükséges az adatok minőségének, életszerűségének ellenőrzése, ugyanis az ujjlenyomat módszernek nagyobb felhasználói bázison is működnie kell. Az összehasonlítás alapja a Panopticlick adatbázis bejegyzéseinek attribútumonkénti entrópiája – amely publikusan elérhető oldalukon. Az összehasonlítás a 6. táblázatban látható. Az *univerzális betűtípusok* és *betűtípusok száma* mezőket később, az adatbázis elemzése során vettem fel. (Az eredmények összehasonlítása során érdemes figyelembe venni, hogy a Panopticlick adatbázisában nagyságrendekkel több bejegyzés van.[43])

Attribútum	Saját entrópia	Panopticlick entrópia
User Agent String	8.095	10.0
Időzóna	2.22	3.04
User ID	9.03	-
Teljes betűtípus lista	8.57	13.9
Univerzális betűtípusok	6.83	-
Betűtípusok száma	7.63	-
Pluginlista	-	15.4

6. táblázat - Adatok entrópiájának összehasonlítása a Panopticlick adatbázisával [17]

Két magas entrópiával rendelkező adatot érdemes kiemelni (a betűkészleteken kívül): egyik az általunk létrehozott *user ID*, másik a Panopticlick által gyűjtött *pluginlista*. A user ID konstrukciója egyedibb értéket ad, mint a teljes betűtípus lista, amiben csupa böngészőfüggő betű is szerepel. Az entrópia tovább növelhető, ha bővítjük egy hosszabb, ritkább betűtípusokból álló listával. A pluginlista entrópiája kiemelkedő, viszont közvetlen felhasználása böngészőfüggővé tenné az azonosítót – és a módszert is. E teszt során nem került sor pluginlista tárolására, így nem volt lehetőség ennek kutatására. (A következő tesztben viszont utólagos elemzés céljából többféle módon is tárolásra kerül.)

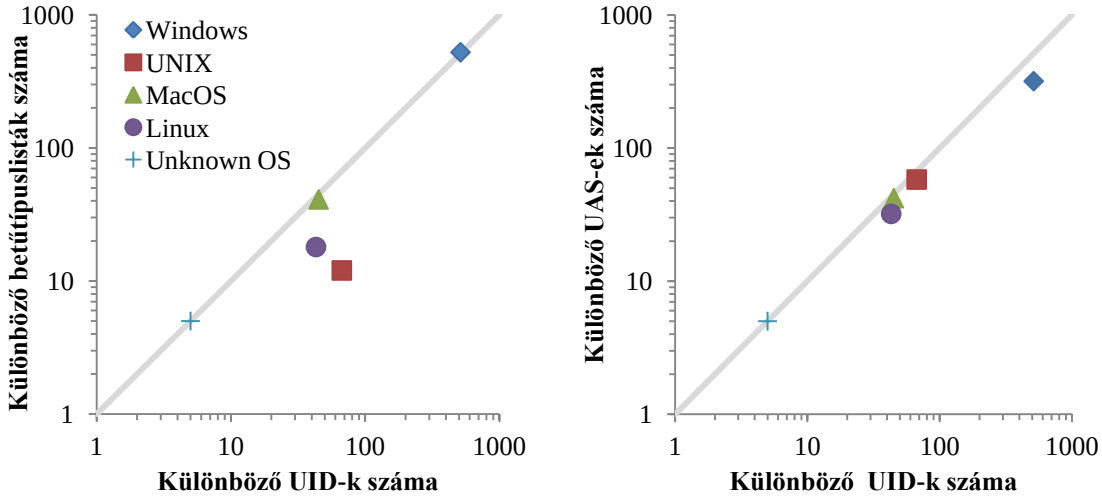
A gyűjtött adatok elemzése

Az adatok elemzésével részletes képet kapunk a komponensekről, továbbá útmutatást nyújt a módszer továbbfejlesztéshez.

[43] Az összehasonlítás ideje 2011. április-május, ujjlenyomat adatbázisukban 1 043 426 bejegyzés volt, ami 470,161 különböző ujjlenyomatot tartalmazott.

4.1.4 Az ujjlenyomatmódszer képességei

Operációs rendszer szerinti összehasonlításnál érdekes módon korreláció mutatkozott a betűtípuslisták és UAS-ek között: az eredmények alapján a betűtípuslisták szilárd alapot biztosítanak az egyedi azonosításhoz Windows és Mac rendszerek esetén (5. ábra bal oldali diagram). Egy másik fontos felfedezés a UAS-ek és a generált felhasználó azonosítók korrelációja – amellett, hogy utóbbi nem tartalmazza egy elemét sem a UAS-nek. Ha azzal a feltételezéssel élünk, hogy a UAS-ek többé-kevésbé egyediek felhasználónként, akkor a generált azonosítók is feltételezhetően pontos azonosítóként működnek minden operációs rendszeren (5. ábra jobb oldali diagram).



5. ábra - Korreláció a betűtípus listák, UAS-ek, és generált felhasználó azonosítók között, különböző operációs rendszerek szerint [17]

Úgy tűnik, hogy a Panopticlick algoritmusához hasonlóan a generált azonosítóval hatékonyan követni lehet a dinamikus IP cím változásokat[44], valamint az is megállapítható, ha különböző számítógépek NAT-on keresztül csatlakoznak[45]. A Panopticlick módszerrel ellentétben az ujjlenyomat hibatűrőnek tekinthető a böngésző- és rendszerfrissítésekkel, több böngésző használatával, pluginlista változásával[46], UAS letiltásával, és természetesen a tárolók ürítésével szemben.

[44] Ekkor csak az IP cím harmadik és/vagy negyedik bájtja változik, ami nem befolyásolja a generált azonosító értékét.
[45] Ebben az esetben az IP cím megegyezik, viszont a rendszerbeállítások, betűtípuslisták eltérőek.
[46] Három esetben módosulhat a pluginlista: új plugin hozzáadásával, plugin eltávolításával, és plugin letiltásával.

Ezen túl számos fontos tényre fény derült az analízis során, felfedeztem több rendszer- és böngésző specifikus gyengeség is. Operációs rendszerre vonatkozólag pl. a .NET keretrendszer verziószámának részletessége, és a nagymennyiségű különböző betűtípus Windows alatt, Linux rendszerek esetén pedig részletes OS verziószám (ld. 7. táblázat). Néhány böngésző-specifikus gyengeség szintén megkönnyíti az ujjlenyomatozást, a részletes „build" kódok és eszköztár verziószámok a legjellemzőbb példák erre.

A UAS elemzésével megismertem az összetevőinek tulajdonságait, és megbizonyosodtam róla, hogy sok felhasználó esetében egyedi azonosító jellege van (kapcsolódó fejezet: A User Agent String elemzése). Meghatároztam az egyes böngésző verziókhoz a UAS egyediségének mértékét, ez később az elemzés során hasznosítható. A betűtípusliták a böngészőprogramot azonosítják – nem böngészőfüggetlenek –, ezért be kellett vezetni az univerzális betűtípuslistát,[47] ezzel erősítve az ujjlenyomatmódszert (ld. 4.1.10 fejezet).

A különféle azonosítók által (UID, UAS, IP cím és betűtípuslista) az ujjlenyomatbeli változások nyomon követhetők, ez annak köszönhető, hogy köztük szinte nincs korreláció. Például, ha ezek közül egy megváltozik, miközben a másik három változatlan marad, akkor nagy a valószínűsége, hogy a két ujjlenyomat ugyanahhoz a felhasználóhoz tartozik; ez lehetőséget ad az ujjlenyomatok felhasználónkénti csoportosítására.

4.1.5 Speciális esetek

Az ujjlenyomatozás robusztussága lehetővé teszi az azonosítási hibák javítását bizonyos esetekben. Egyszerű lekérdezések segítségével sikerült azonosítanom néhány visszatérő látogatót, akik

- egynél több böngészőt is kipróbáltak,
- többféle képernyőfelbontás és időzóna kombinációt használtak,
- vagy dinamikus IP címük megváltozott.

A különböző attribútumok összehasonlítása mellett a tárolt UAS-ek is segítettek az azonos felhasználóhoz tartozó rekordok párosításában. E megfigyelések alapján az attribútumok változása követhető, bizonyos korlátok mellett. A követési technikát akár automatizálni is lehet, de ez túlmutat jelen elemzésen. (De nem is lehetetlen: a

[47] Adott rendszeren különböző böngészőprogramokból azonosan elérhető (megjeleníthető) betűtípusok összeválogatásával.

Panopticlick projektben kidolgoztak egy egyszerű javítási technikát, ami lehetővé teszi az ujjlenyomat változások pontos követését.)

Voltak esetek, amikor az algoritmus hibázott, három különböző típusú hibára lettem figyelmes. Első esetben a felhasználó különböző azonosítókat kapott, amikor többféle böngészőprogrammal próbálkozott. Összesen 28 ilyen eset volt, mindet a válogatott betűtípuslista okozta, mint kiderült – szemben az előzetes feltételezéssel – nem minden betűtípus böngészőfüggetlen. Az univerzális betűtípuslista bevezetése a jelenség előfordulását 6 ujjlenyomatra csökkentette, ennek a számnak a további csökkentését a második hiba megoldásával lehetett orvosolni.

A második hibaesetben annak ellenére, hogy a vizsgált rekordokban minden tényező megegyezett, a betűtípuslista eltérő volt (és egy esetben az OS is), aminek eredményeképp az algoritmus eltérő azonosítókat generált. Ez 6 bejegyzés esetében fordult elő, amelyből 4-nél azonos volt a UAS, 5 közülük Linux, egy pedig Windows 98 rendszerről érkezett (állítólag). Ez nem feltétlenül jelent hibát, ugyanis a felhasználó telepíthetett (vagy törölhetett) betűtípusokat a tesztek elvégzése között, és vannak eszközök a UAS hamisítására is.

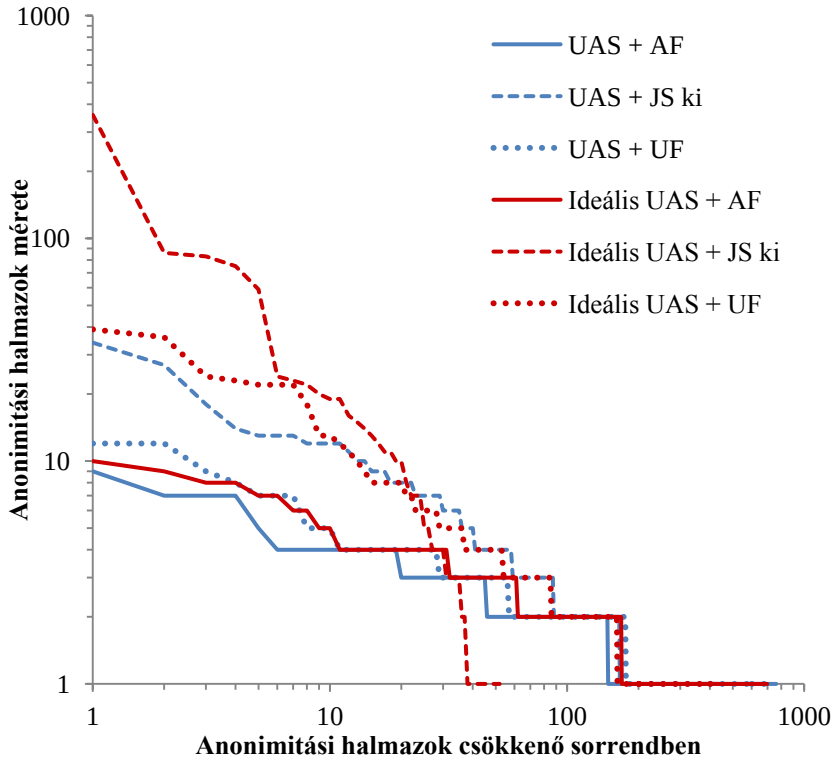
A harmadik típusú hiba esetében minden egyes attribútum azonos volt a vizsgált rekordokban (beleértve a UAS-et és a teljes fontlistát is), mégis különböző azonosítókat generált a rendszer. Ez 5 pár rekordnál fordult elő, a hibát nem sikerült reprodukálni, eredete ismeretlen.

4.1.6 Anonimitási halmazok

A különböző azonosítók értékeinek halmaza felbontható anonimitási halmazokra. A fejezet témája a UAS-ek és a betűtípuslisták anonimitási halmazainak növelhetőségének tanulmányozása a jelenlegi feltételek mellett. Sajnos minden egyes azonosítóra igaz, hogy sok kis méretű halmazra bontható fel – az anonimitás elérése tehát nehéz feladatnak tűnik. Az ábrákon AF (Arbitrary Fonts) jelzi, amikor a felhasználó bármilyen tetszőleges fontot használhat (a valós állapot), és UF (Unified Fonts) jelzi, amikor minden felhasználó ugyanazt az egységes betűtípuslistát használja[48], ami ez által nem minősül megkülönböztető azonosítónak[49] (ideális állapot).

[48] Ez egy felvetés, miszerint az összes böngészőprogram egységes, előre definiált betűtípusokat használna, és azokat ugyanúgy képes lenne megjeleníteni. (A betűtípusok számossága ez esetben nem lényeges, a hangsúly az egységességen van.)
[49] A többi attribútum természetesen ugyanúgy elérhető JS-en keresztül.

Az elemzés idején használt UAS-nek több, túlz részletes verziószámmal ellátott komponense van, ami nagymértékben csökkenthető lenne, ugyanis nincs rájuk szükség. Ennek okán „ideális UAS” a UAS-nek egy olyan változata, amelyben nincsenek részletes verziószámok, csak ami mindenképp szükséges. A 6. ábrán látható, hogy ideális UAS mellett sokkal nagyobb anonimitási halmazok lennének, mint a valódi UAS esetén. Meglepő eredmény, hogy az ideális UAS és UF (JS detektálás tehát aktív) együttes alkalmazása nagyobb anonimitási halmazokhoz vezet, mint a valódi UAS a JS teljes kikapcsolása mellett.



6. ábra - Anonimitási halmazok alakulása valós és ideális UAS esetén, különböző paraméterek mellett. [17]

Az ideális UAS főként Linux felhasználók esetén lenne fontos, hiszen sokkal nagyobb anonimitási halmazok kialakítása lehetséges, hogyha a UAS nem tartalmazza a nagyon részletes operációs rendszer verziószámot (ld. a UAS elemzésnél). Bekapcsolt JS-tel a helyzet még rosszabb, az első 11 anonimitási halmaz után csak egyedi felhasználók vannak.

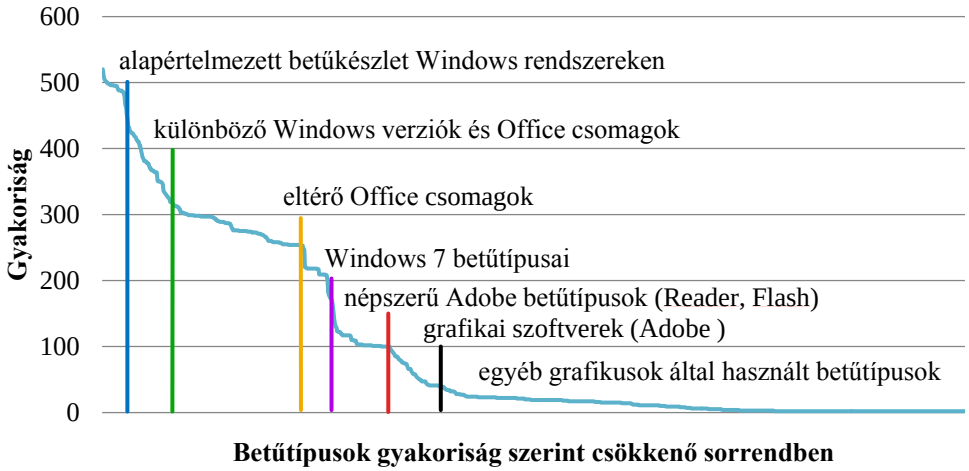
4.1.7 Betűtípuslisták

Fontos megjegyezni, hogy az összeállított betűtípus adatbázis tartalmaz speciális, grafikusok által használt fontokat is, nem csak az elterjedteket. Ezekkel is volt néhány teszt, ami befolyásolhatja az eredményeket.

Az elemzés előtt feltételezhető volt, hogy a betűtípus listák összefüggésben vannak az operációs rendszerrel, ennek bizonyítására készítettem egy speciális lekérdezést. Az eredmények alapján Windows és Mac esetén a különböző betűtípuslisták száma közel van a különböző felhasználók számához (Windows esetén meg is haladja), azonban Unix alapú rendszerek esetén a különböző betűtípuslisták száma sokkal alacsonyabb. Ez arra utalna, hogy Unix alapú rendszereknél a betűtípuslista nem annyira releváns tényező, mint a többi rendszer esetében. A második teszt előkészületeinél azonban fény derült rá, hogy bizonyos népszerű Linux verziók esetén hibás eredményt ad a JS alapú betűtípus detektáló, mivel az ilyen rendszereken végzett tesztek 70%-át érinti, ezért a feltételezést ezzel nem sikerült bizonyítani. Utólagos vizsgálatok alapján – a második teszt adatait használva – bizonyítható a feltételezés, Windows és Mac esetén valóban több az egyedi betűtípuslisták száma. Mindemellett Linux alapú rendszereknél is jelentős az arány (60%), tehát a betűkészlet detektálás mindenképp releváns része kell hogy legyen az ujjlenyomatoknak.

Ábrázoltam a tesztelők által használt különböző betűtípusok gyakoriságát csökkenő sorrendbe, az eredmény egy lépcsőzetes görbe (ld. 7. ábra). Feltételeztem, hogy az egyes lépcsőfokok bizonyos szoftverek használatára utalnak. A fontok forrásának (szállító szoftver) felderítésével[50] meggyőződtem róla, hogy a görbe fokozatai bizonyos szoftverek (ill. szoftvercsomagok, operációs rendszerek) használatát jelzik, amelyek telepítéskor bővítik a rendszerre telepített betűtípusok listáját.

[50] A Microsoft és az Adobe is készített publikusan elérhető listát a szoftvereikkel szállított betűtípusokról, a betűtípusok többségének forrását ezek alapján sikerült beazonosítani.



7. ábra - Betűtípusok gyakorisága csökkenő sorrendben Windows rendszeren [17]

A több böngészőt is kipróbáló felhasználóknál sok esetben eltérés volt a detektált betűtípuslistában. Ezt a különböző böngésző implementációk okozzák, másképp kezelik a betűtípusokat. A betűtípus detektáló script saját gépen való futtatásával több különböző esetet sikerült megfigyelnem. A Firefox betűcsalád helyettesítést alkalmaz: hogyha hiányzik egy font, akkor azt egy azonos családból származó másik fonttal helyettesíti (pl. ha Helvetica nem elérhető, akkor Arialt használ). Ezzel szemben az Opera nem használ helyettesítést, sőt, fel sem ismeri az összes telepített fontot.

Ez azt jelenti, hogy a betűtípus detektálás nem böngészőfüggetlen technika a teljes betűtípus adatbázis használata esetén. A probléma megoldásához készíteni kellett egy új listát, amelyben csupa olyan font szerepel, ami az öt fő böngészőből[51] azonosan felismerhető. A lista az „univerzális betűtípuslista” nevet kapta, a fontokat pedig a szerint válogattam bele, hogy kilistáztam a legtöbb különböző böngészőből elérhető fontot böngésző gyakoriság szerint csökkenő sorrendben, és betűtípus próbálgatás alapján meghúztam egy határt ott, ameddig függetlennek mutatkozott. Ez természetesen csak egy demonstráció, nem feltétlenül hibátlan a lista, hiszen nem követeltem meg, hogy mind az öt böngészőből legyen képviselő, ehhez sokkal több ujjlenyomattal kellene rendelkezni (legtöbben csak kétféle böngészővel teszteltek egy rendszerből).

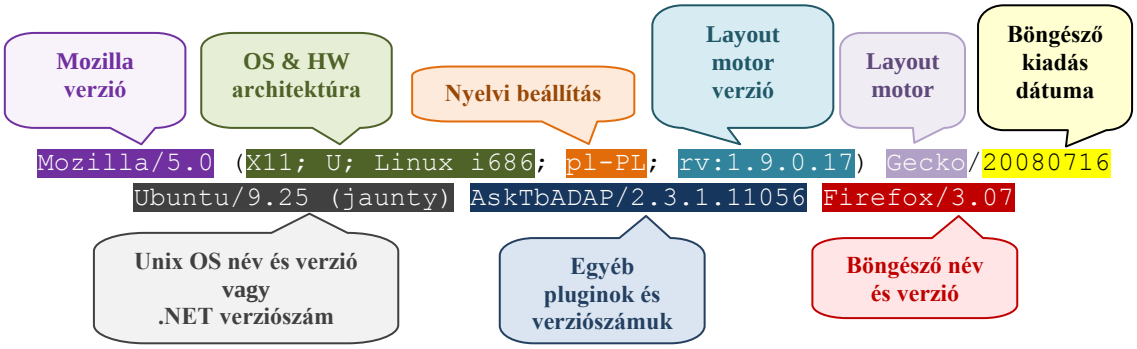
Peter Eckersley javaslata szerint a böngészőknek csak megerősíteni kellene egy betűtípus vagy plugin létezését a listaszerű felsorolás helyett, valamint a lekérdezések válaszát exponenciális késleltetéssel kéne ellátni, megelőzve ezzel a rosszindulatú

[51] Internet Explorer, Firefox, Chrome, Safari és Opera

kódok által alkalmazott kimerítő keresést. Azonban a JS betűtípus detektáló módszer emellett is működőképes maradna, hiszen az *offsetWidth* és *offsetHeight* DOM attribútumok változását kérdezi le különböző betűtípusok esetén, nem pedig a font létezését, tehát hatásos védekezés ezen attribútumok eltávolítása a JS API-ból, vagy a használható fontok számának korlátozása valamilyen módon.

4.1.8 A User Agent String elemzése

A User Agent String struktúrája indokolatlanul komplex volt a kísérlet idején (8. ábra), a sok részletes komponens miatt a kialakítható kombinációk száma nagyon nagy. Részletességét még tovább növelik bizonyos pluginok és eszköztárak, amelyek telepítéskor hozzáfűzik saját nevüket és verziószámukat.



8. ábra - A UAS struktúrája [17]

A Firefox 3.6 az esetek közel 40%-ában különböző UAS-gel rendelkezik – ennek is a 70%-a eltérő felhasználókhoz köthető. Windows rendszerek esetén lényeges megkülönböztető tényező a .NET keretrendszer verziószáma, mert nagyon részletes. Ugyanaz a verzió is képes eltérést okozni a megjelenésben: egyik esetben bekerül egy szóköz a pont elé, másik esetben több telepített verzió is megjelenik, ld. 7. táblázat. Ezen túl vannak esetek, amikor két UAS a .NET verziószámig teljesen megegyezik, azonban áll utána még egy plugin vagy eszköztár bejegyzés is, ami miatt mégis különbözők lesznek.

A böngésző kiadások dátumai szintén megkülönböztető erővel bírnak. Az adatgyűjtés ideje alatt Operából olyan gyakran készült új kiadás, hogy sok esetben maga a verziószám egyedivé teszi a böngészőt. Hasonlóképpen a Linux rendszerek felhasználóinak közel 70%-a egyedi a disztribúció neve és száma miatt, ami relatíve gyakran változik. E két esetben fontos tényező, hogy kevesek által használt szoftverekről van szó, ami még inkább az egyedivé teszi a felhasználót. Sok különféle

UAS variációt hozott létre a disztribúció verzió, kiadási dátum, layout engine verzió, plugin és eszköztár verziók kombinációja.

A UAS könnyen lerövidíthető kevésbé részletes verziószámok használatával, ezzel csökkentve az egyediség valószínűségét. (A Panopticlick projektnek is van egy hasonló javaslata.) Például elég lenne csak az OS család neve a teljes verziószám nélkül – főként Linux esetében. Hasonlóképpen elegendő lenne a fő böngésző verziószám megjelenítése, ami egy vagy két számjegyből áll. A kiadás dátumára, layout engine-re és pluginokra vonatkozó részek véleményem szerint teljesen elhagyhatók, feleslegesek. Az ideális UAS az alábbi összetevőket tartalmazná: OS család, nyelvi beállítás, böngésző neve és fő verziószáma, pl. „Opera/10 (Ubuntu; hu)" – ez minden információt tartalmaz, amire egy webes szolgáltatónak szüksége lehet a megjelenítés optimalizálásához vagy statisztika készítéséhez.

Az emberek többsége egyszerre csak egy böngészőprogramot használ, azt is többnyire hónapokig. Ebből adódik, hogy több felhasználó egyazon IP cím mögül csatlakozva (pl. otthoni vagy irodai hálózat) megkülönböztethető már a UAS alapján is, hogyha nem ugyanazt a böngészőverziót használják (és ugyanazt az OS verziót, eszköztárakat, és .NET verziót). A UAS tehát lokális azonosítóként használható, mindazonáltal vannak esetek, amikor globálisan egyedinek tűnik (7. táblázat). Meg kell említenem, hogy dolgozatom írásának idejére már a UAS-ek egyszerűsödtek, jelentősen kevesebb információt tartalmaznak, mint az adatgyűjtési időszakban szerzett példányok. Ez pozitív előrelépés, viszont a verziószámok még mindig túl részletesek.

.NET verzió	Eszköztár verzió	Kiadás dátuma	Böngésző verzió	Linux verzió
(.NET CLR 3.0.04506.648)	GTB7.1	20100513	9.0.597.15	Ubuntu/10.10 (maverick)
(.NET CLR 3.5.30729)	GTB7.0	20100611	9.0.597.19	Ubuntu/9.10 (karmic)
(.NET CLR 3.5.30729)	AskTbPLTV 5/3.8.0.12304	20100722	9.0.597.44	SUSE/3.6.12-0.7.1

7. táblázat - Példák egyedi UAS komponensekre[52] [17]

4.1.9 Tesztek: Privát böngészési mód és anonim böngészés

TOR, vagy bármi más proxy használata az ujjlenyomatozó módszer ellen hatástalan. Ezek a technikák csak a szolgáltató által látott IP címet változtatják meg, a

[52] Az adatok a felhasználók privát szférájának védelme érdekében meg vannak keverve, ill. egy részük a http://useragentstring.com weboldalról származik.

többi paraméter továbbra is jól használható nyomkövetésre. Ennek igazolására speciális lekérdezéseket futtattam, és találtam 9 pár rekordot, ahol páronként minden attribútum értéke megegyezett, kivéve az IP címet. A közeli időbélyegek arra utalnak, hogy a felhasználó nem változtathatta meg jelentősen a helyzetét, tehát azonos végpontról csatlakozott, de valamilyen hálózati anonimizáló szolgáltatáson keresztül[53], vagy dinamikus IP cím megújítással. Két esetben a felhasználó az IP címen kívül még egy paramétert módosított, de az IP cím első két bájtja páronként megegyezik (feltehetően dinamikus IP cím megújítás miatt), bizonyára ugyanarról a felhasználóról van szó. Egy esetben okkal feltételezhető, hogy a felhasználó megváltoztatta helyzetét, hiszen nagy a különbség (több mint egy óra) a két időbélyeg között, lehetséges, hogy laptopról csatlakozott két különböző helyről.

Általánosságban javasolt a sütik tiltása (vagy rendszeres törlése) ajánlott, de hiábavaló az ujjlenyomat alapú nyomkövetés ellen. Privát böngészési mód használatakor a böngészőprogram csak a memóriában tárolja az előzményeket, gyorsítótár, sütik és egyéb tárolók tartalmait – nem írja ki a háttértárra –, ezáltal azok törlődnek a privát böngészés befejezésekor, nem hagynak nyomot. Mindazonáltal ez nem véd a passzív (perzisztens tárolás nélküli) nyomkövetés ellen, hiszen nem tiltja le a JavaScriptet, és az IP cím és UAS továbbra is látható. Az ujjlenyomat módszer mindaddig működni fog privát módban is, amíg a gyártók nem térnek át az egységes, kevésbé részletes komponensek alkalmazására.

Modern böngészőprogramok adatvédelmi beállításainál a felhasználó megjelölheti, hogy szeretné elkerülni a nyomkövetést (Do Not Track[54]). Ám a beállítás figyelembevételére a szolgáltató nincs kötelezve, saját döntése, hogy tiszteletben tartja-e a felhasználó kérését, vagy sem. Amennyiben nem, a kérés ellenére továbbra is folytatódhat a nyomkövetés, hasznossága a jövőben tehát azon múlik, hogy a szolgáltatók hajlandók-e széles körben alkalmazni a koncepciót.

4.1.10 Az ujjlenyomatozó módszer robusztusságának növelése

Az elérhető konfigurációs beállítások (pl. képernyőfelbontás) a rendszer telepítésekor jönnek létre, és azt követően módosításukra nagyon ritkán kerül sor – általában soha. Az értékek változatossága miatt a generált felhasználó azonosító

[53] Természetesen nem zárhatjuk ki, hogy valójában két különböző felhasználóról van szó, akiknek betűtípusra pontosan megegyezik a konfigurációja, ám ennek valószínűsége kicsi.
[54] "Do Not Track" Explained: http://33bits.org/2010/09/20/do-not-track-explained/

rendelkezik a legnagyobb entrópiával az ujjlenyomat adatbázisban. Dinamikus IP címváltozás nem okoz problémát, mert csak az első két bájtot tartalmazza az ujjlenyomat, azonban a tartósnak feltételezett beállítások módosítása (képernyőfelbontás, időzóna, és alap betűkészlet) új azonosító generálásához vezet. A változások detektálhatók a változatlanul maradt adatok alapján (IP cím, teljes fontlista, és UAS) – bizonyos korlátokkal.

Plugin detektálás hozzáadásával szintén növelhető az ujjlenyomat algoritmus robusztussága, hiszen sokféle plugin létezik, a verziószámok pedig részletesek, ezáltal a kombinációk száma nagyon magas. Ezt megerősíti a Panopticlick projekt jelentésében szereplő kiemelkedő entrópia érték is, tehát érdemes egy böngészőfüggetlen plugin detektálási eljárást kifejleszteni, hasonlóan, mint betűtípusok esetén. Azonban körültekintően kell összeválogatni a kiegészítőket, érdemes a ritkábban frissülőket bevenni, vagy gyakran frissülők esetében verziószám csonkolást végezni – legalkalmasabbak a Java, Flash és Silverlight pluginok lehetnek.

4.1.11 Hogyan tovább?

Fő probléma a kísérlettel a rekordok mesterséges összeköthetősége, ezért létre kell hozni egy független azonosítót (pl. becenév vagy LSO süti), amellyel a rekordok csoportosíthatók felhasználók szerint, ez sokat segíthet a későbbi elemzés során (hiszen globális azonosító hiányában gyakran feltételezésekre kell építeni). Mivel nagy volt az egy tesztet végzők száma, fel kell hívni a felhasználók figyelmét, hogy próbálják ki több böngészővel a tesztet. Ezen túl érdemes eltárolni a ténylegesen használt betűtípus listát (Java vagy Flash detektálás), ezáltal árnyaltabb képet kapunk a különböző rendszerekre telepített betűtípusokról, és javítható az univerzális fontlista is. Későbbi analízis céljából a plugindetektálás is része lesz a tesztnek – alkalmazásával megismerhető a pluginlista karakterisztikája, amely segít eldönteni, hogy érdemes-e hozzávenni a felhasználó azonosítóhoz.

A fent említett javaslat mellett több módja is van az algoritmus javításának, például egy hibrid technika (ujjlenyomatozás kombinálása perzisztens tárolással) növelheti a módszer robusztusságát. A generált azonosító tárolható pl. evercookie technikával (de még jobb lehet egy független véletlen szám), melynek segítségével ténylegesen nyomon követhető az azonosító változása[55]. A perzisztens tárolás mellett

[55] Például ha rendszerbeállítás változása, vagy betűtípus telepítés következtében megváltozik az azonosító.

szól még egy érv: a vállalati rendszerekben, ahol a számítógépek konfigurációja gyakran ugyanaz (központi telepítés), a passzív módszerek nem hatékonyak, az egyes gépeket csak perzisztens azonosítóval[56] lehet megkülönböztetni.

Annak érdekében, hogy részletesebb képet kapjunk az anonimitási halmazokról, nagyobb mintahalmazra van szükség, sok különböző eszközzel és rendszerrel. A mobil eszközök nagyon gyors ütemben terjednek, érdekes lehet az ujjlenyomat módszer hatékonyságának megfigyelése ebben a környezetben. A nemzetközi látogatók számának növelése sokat segítene ezekben, hiszen külföldön elterjedtebb a Mac és Unix alapú rendszerek használata, és a mobil eszközök is változatosabbak.

Értékelés

A kutatás eredményei rávilágítottak, hogy a rendszerujjlenyomat módszer egy valóságban is működő, jól alkalmazható nyomkövetési technika. A kutatás eredményeiről 2011 nyarán konzulensemmel és kollégáival közösen írtunk egy angol nyelvű cikket [17], amit beadtunk a *Nordsec 2011* nevű információs technológiák biztonságával foglalkozó konferenciára. Publikációnkat elfogadták, a Springer kiadó weboldalán elérhető elektronikus formában[57], egy draft változata pedig letölthető a Böngészőfüggetlen ujjlenyomat teszt 2.0 weboldaláról[58].

[56] Ebben a környezetben fontos, hogy a perzisztens azonosító ne kizárólag a kliens beállításai alapján jöjjön létre (így könnyen feleslegessé válhat), hanem tartalmazzon valamiféle véletlen, vagy időfüggő komponenst.
[57] http://www.springerlink.com/content/4775110ql0h17844/
[58] http://fingerprint.pet-portal.eu/index.php?menu=2

5 Böngészőfüggetlen ujjlenyomat teszt 2.0

Az első tesztből szerzett ismeretek, és a kimerítő elemzés eredményei megfelelő alapot adtak a teszt továbbfejlesztésére, és egy újabb, nagyobb szabású adatgyűjtés indítására. A projektben fontos szerepet kap a felhasználók privátszféra tudatosságának növelése is, ennek szellemében készült az új adatgyűjtő weboldal és maga az ujjlenyomatozó mechanizmus is.

Előkészítés

A fejezet témája a célok specifikálása a második teszt eredményei alapján, és az ehhez szükséges módosítások megfogalmazása. Az első adatgyűjtés által létrehozott adatbázis vizsgálata során kiderült, hogy kis minta esetén nagymértékű az egyediség, bizonyos esetekben akár egyetlen attribútum is elegendő az azonosításhoz; azonban nem derült ki, hogy a módszer hatékony-e nagymennyiségű felhasználó – és nagymennyiségű hasonló adat – esetén.

Utóbbinak több befolyásoló tényezője is van, egyik az, hogy mennyire széles skálán mozog a látogatók tábora. Lényeges, hogy az egyszerű felhasználók legyenek túlnyomó többségben, de szükség van szakértőkre is, akik szándékosan megpróbálnak védekezni a követési módszer ellen, különböző eszközök bevetésével. Fontos az ujjlenyomatozó technika hatékonysága is, azaz lehetséges-e minden felhasználó megkülönböztetésére alkalmas, de mégis böngészőfüggetlen azonosító létrehozása (böngészőfüggő megoldásra jó példa a Panopticlick, de a cél továbbra is a böngészőfüggetlenség). Ez a másik oldalról tekintve is megfogalmazható: széleskörű felhasználói forgalom mellett is lesz-e a felhasználók között egy-egy eltérő adatmorzsa, amely lehetővé teszi a megkülönböztetést, vagy a nagy halmazok kialakulása miatt csak böngészőfüggő, specializáló megoldás lehetséges. Az előző teszt során ez sajnos nem derült ki, mert a tesztelők jelentős része (72%) csak egyetlen tesztet végzett, fontos szerepet kap tehát ennek igazolása.

Az új teszt egy dologban lényegesen eltér elődjétől: a passzív ujjlenyomatozó eljárás mellett aktív technikát is alkalmaz – illetve technikák gyűjteményét –, a tesztbe beépítettem az Evercookie perzisztens tárolót is. Az ujjlenyomatozás működését a

perzisztens tárolás nem befolyásolja,[59] ugyanakkor megfigyelhetővé teszi az ujjlenyomatok (és az összes rögzített adat) változását, hiszen egy független azonosító segítségével precízebb az egy felhasználóhoz tartozó bejegyzések csoportosítása.

5.1.1 Célok

Természetesen továbbra is a böngésző- és pluginfüggetlen rendszerazonosító létrehozása az elsődleges cél, annak továbbfejlesztése és robusztusabbá tétele, illetve az utófeldolgozáshoz és analízishez szükséges adatok előállítása mellett.

Új célok az összegyűjtött adatok alapján:

- pluginlisták elemzése,
- Linux és Unix rendszereken használt böngészőfüggetlen betűtípusok gyűjtése (későbbi beépítés céljából),
- az esetlegesen hibásan detektálható betűtípusok kiszűrése,
- a megváltozott összetételű UAS elemzése,
- a HTTP Accept fejlécek elemzése,
- a böngészőprogram és a rendszer együttes egyediségének elemzése,
- és nem utolsó sorban az együtt alkotott hibrid módszer elemzése.

5.1.2 Javítások a rendszerujjlenyomaton

A továbbfejlesztés első lépése az előző módszer problémáinak javítása, a bővítés csak ez után következhet. Alapvetően három módosítás szükséges, ezek közül első és legfontosabb az alap betűtípuslista (basic fonts) újraválogatása, második az IP cím kétbájtos összetevőjének kivétele, harmadik pedig a bejegyzések explicit nyomkövethetőségének és csoportosíthatóságának fejlesztése.

5.1.2.1 Az alap betűtípuslista újraválogatása

Az alap betűtípuslista a generált azonosító legfontosabb alkotóeleme, hiszen a legnagyobb értékkészlettel bír a komponensek között, szinte egyedileg képes azonosítani egy felhasználót. Összetételének meghatározása ezért kritikus, és egyben nehéz feladat, a betűtípusok több jellemzőjének ismerete is szükséges hozzá. Ezek közé tartozik elsősorban az operációs rendszer- és böngészőfüggetlenség, illetve előfordulásuk gyakorisága (vagyis a megkülönböztető érték). Az újraválogatás legfőbb célja a böngészőfüggetlenséget megsértő betűtípusok eltávolítása. Az operációs

[59] Nem a helytelenül generált azonosító átírása a cél, hanem a változások követése.

rendszer függetlenség megvalósítható rendszer specifikus egyedi lista összeállításával, vagy rendszerfüggő fontdetektálással. A másik lehetőség egy univerzális betűtípuslista használata, ez esetben teljesülnie kell az operációs rendszerenkénti böngészőfüggetlenségnek – munkámban ezt a megoldást választottam. Lényeges emellett, hogy ne csak az operációs rendszerrel telepített betűtípusokat tartalmazza a lista, mert ezek nagyon gyakoriak, nincs releváns megkülönböztető értékük. Tartalmazzon különböző szoftverek (vagy szoftvercsomagok) által szállított betűtípuscsoportokat, hiszen az alapvető eltérést a rendszerek között ezzel lehet leghatékonyabban reprezentálni.

A betűtípus válogatás több fázisban zajlott. Először Windowsra jellemző programokra bontva válogattam (a teljesség igénye nélkül), ezt követték a Linux és Unix betűtípusok, végül kis késleltetéssel a Mac betűtípusok. Windows esetében könnyebb volt a feladat, hiszen a Microsoft oldalairól publikusan elérhetők az Office csomagokkal szállított betűtípusok, ugyanez jellemző az Adobe termékekre, és nem utolsó sorban az első teszt által gyűjtött ritka, de több böngészőből elérhető betűtípusok listája is rendelkezésemre állt. A többi rendszer esetén jóval nehezebb a feladat, hiszen a Linux betűtípus listák nagy része használhatatlan (ld. 4.1.7 fejezet), a Mac-kel létrehozott listákban pedig kevés olyan volt, amely az egyedi és böngészőfüggetlen kritériumnak is megfelel. A válogatás tehát e két esetben algoritmikus úton nem lehetséges, ezért más módszert kellett találni: virtualizált környezetben telepítettem az operációs rendszerek egy-egy példányát, és a teszthez szükséges böngészőprogramokat. Ezek után kereséssel letöltöttem egyedinek feltételezett ingyenesen elérhető betűtípusokat, amiket kipróbáltam a különböző rendszer-böngésző kombinációkon. Sajnos a folyamat lassú, és hatásfoka is alacsony, a kipróbált fontok közül kevés felelt meg a kritériumoknak.

Eredményképp az új alap betűtípuslista az előző hosszának körülbelül duplája lett, kerültek bele szoftver specifikus betűtípusok (Windows), és több Linux és Mac font, amik feltételezhetően több felhasználót tehetnek egyedivé a böngészőfüggetlenség megtartása mellett. A jövőben ennek továbbfejlesztését (ismételt újraválogatás) mindenképp algoritmikus úton – gyűjtött adatok alapján – érdemes végezni, ezért az előző adatgyűjtésből kimaradt tényleges betűtípus lista lekérdezését (Flash módszerrel) is tartalmazza az új teszt. A következő újraválogatás ez által sokkal nagyobb és precízebb listát eredményez, a művelet hatékonyabb és gyorsabb lesz. Kiemelt szerepet kap emiatt is, hogy a felhasználók minél több böngészővel végezzék el a tesztet.

5.1.2.2 IP cím összetevő eltávolítása

Az előző tesztben használt ujjlenyomat tartalmazza az IP cím első két bájtját. Ez azonban lekorlátozza a generált azonosító hatékonyságát, ugyanis csak dinamikus IP címváltásnál marad meg az érétke, amint hálózatot vált a felhasználó (pl. wifi, tor vagy proxy használatával), megváltozik az azonosító is. Ezáltal nehezebbé válik a követés, figyelembe kell venni a többi attribútumot is, ami ha nincs automatizálva csak utólag csoportosíthatók a rekordok, és a rendszerazonosító sem lesz teljes értékű, hiszen hálózatfüggő. Elhagyása az azonosítóból tehát ésszerű döntés, mindemellett az automatizáláshoz és utófeldolgozáshoz szükséges funkcionalitása megmarad, hogyha tároljuk.

5.1.2.3 Bejegyzések explicit csoportosítása

Erre az utófeldolgozás során lesz szükség, illetve az 5.1.3.4 fejezetben tárgyalt kezdetleges ujjlenyomat követő algoritmusban. Az ujjlenyomatok változásának hatékony követése – társítása korábbi bejegyzésekhez, felhasználóhoz – nem triviális feladat, és nincs is rá hibamentes megoldás. A korábbi analizálás során erre a célra *konstansnak vélt* attribútumok lettek felhasználva, ami a kis minta miatt hatékonynak bizonyult, de sok felhasználó esetén nagyobb a kockázata a téves párosításnak – nagyobb eséllyel fordul elő két azonos konfigurációval rendelkező felhasználó.

A legpontosabb feldolgozás érdekében szükséges tehát, hogy a begyűjtött adatoktól függetlenül is párosíthatók legyenek a bejegyzések, ehhez a felhasználók segítségére van szükség. Az első teszt indításakor a látogató megadhat egy becenevet, ami az explicit azonosítója lesz – csak olyan név választható, ami még nem szerepel az adatbázisban. Ez tárolásra kerül Evercookie használatával, ezáltal nincs szükség újbóli megadásra (ami nem is lehetséges, mivel már szerepel az adatbázisban). A becenév bekerül egy böngészőfüggetlen tárolóba is (pl. LSO tár), ezért másik böngészőprogrammal való teszteléskor nem kell újat megadni, az algoritmus kiolvassa azt. Ellenkező esetben a felhasználóra van bízva az új név megadása, célszerűen számozhatja a korábban megadott nevet – prefixként használhatja. A tesztelő a beállított Evercookie tartalmát törölheti egy gomb segítségével. A felhasználó beleegyezése nélkül is eltárolható a böngészőjében egy azonosító erre a célra, azonban ez privát szférát sértő megoldás, ezért választottam más, opcionális módot.

Adatgyűjtés

A módszer hatékonyságának mérésén kívül az adatgyűjtésnek további fontos célja az utófeldolgozáshoz szükséges kutatási adatbázis elkészítése, ezért az első adatgyűjtéshez képest jelentősen több adatot terveztem menteni. Ezáltal az adatlekérő script futásideje és a hálózati forgalom is növekszik, azonban ez nem jelent problémát, hiszen az eljárás nem „ipari alkalmazásra", hanem demonstrációs célra készül. Az első teszt elemzése során létrehozott adatfeldolgozási módszerek egy részét automatizáltam, és beépítettem a mentés folyamatába.

5.1.3 Módosítások az ujjlenyomat adatbázison

Az új adatbázis technikailag két részre bontható, egyik az adatgyűjtő oldal tartalomkezelését, másik pedig az adatgyűjtés folyamatát szolgálja, e fejezet témája a második csoport. A *fingerprints* tábla bővítését a 8. táblázat foglalja össze. A törölt attribútumokat bordóval, az újonnan gyűjtötteket zölddel és kékkel jelöltem, a kék ezen belül a többi adatból származtatott, szerver oldalon számított értékeket jelöli.

Mező	Leírás
id	a rekord egyedi azonosítója
locality	a használt nyelvi beállítás (hungarian/international)
~~user_id~~	~~a generált felhasználó azonosító~~
user_id_short	a felhasználó azonosító hash formában
created	rekord létrehozásának ideje
ip	látogató IP címe titkosított formában
user_agent	User Agent string
build_id	a böngésző kiadási dátuma
dnt	Do Not Track bit
accept_header	négy Accept fejléc egymás után fűzve (lásd 1. táblázat)
os	operációs rendszer
screen	képernyő felbontás
screen_avail	rendelkezésére álló képernyő felbontás[60]
timezone	időzóna
fonts[61]	alap betűtípuslista az első tesztből
fonts_all[62]	betűtípusok listája bináris formában
fonts_count	a detektált betűtípusok száma
fonts_univ	univerzális betűtípuslista (lásd 4.1.7)
fonts_feature	az újraválogatott betűtípuslista
plugins_univ	univerzális pluginlista

[60] A böngésző ablak által ténylegesen kitölthető dimenzió, pl. Windows esetén általában a tálca méretével kisebbített érték.
[61] Az előző tesztben külön listából kliens oldalon történt az előállítása, hexadecimális formában.
[62] Az előző adatbázisban *fonts_bin* néven szerepelt.

Mező	Leírás
plugins_nonu	böngészőfüggő pluginok listája (előző komplementere)
plugins_all	teljes pluginlista
mimetypes	formátumtípusok listája
navigator_hash	a navigator objektumról készített hash
browser_ver	a böngésző neve és fő verziószáma
tracking_id	vélhetően azonos felhasználóhoz tartozó korábban létrehozott rekord azonosítója
passive_id	a *tracking_id*-hoz hasonló, de nem veszi figyelembe a *user_name* értékét
user_name	a felhasználó által megadott, Evercookieból kiolvasott becenév (ha nem adott meg, akkor a generált azonosító)
fire_gloves	a FireGloves kiegészítő használatát jelzi

8. táblázat - A *fingerprints* tábla bővített struktúrája

A felhasználó azonosítóban az új attribútumok közül egyedül a *fonts_feature* szerepel a korábban használt *fonts* helyett, a többi mind utólagos elemzés céljából kerül mentésre. A további alfejezetekben az új adatok külön magyarázatot igénylő lekérdezési és számítási mechanizmusát részletezem.

5.1.3.1 Betűtípuslisták

Összesen négy különböző betűtípuslista kerül tárolásra, azonban kliensoldalon elegendő a négy közül a legnagyobb, azaz a *fonts_all* létrehozása, az összes többi ennek egy speciális részhalmaza, előállíthatók belőle szerver oldalon is. Azért is fontos, hogy szerveroldalon történjen, mert a kliensoldali algoritmus leg erőforrás igényesebb részművelete a JS fontdetektálás, célszerű tehát alkalmazásának minimalizálására törekedni.

A betűtípusok detektálása az első teszttel teljesen azonos módon a *fingerprint_fonts* táblában tárolt betűtípusok alapján történik, eredménye egy binárisan ábrázolt betűtípuslista, ami a többi adattal együtt jut el a szerverre. A *fingerprint_fonts* tábla is ki lett egészítve halmaz attribútumokkal (*basic, univ* és *feature*), adott halmazba tartozás esetén az attribútum értéke `1`, egyébként `0` (ld. 9. táblázat). A JS fontdetektálás csak az eredeti adatbázist használja, a később dinamikusan hozzáadott betűtípusokat nem, hiszen többezres listánál nagyon nagyra nőne a detektálási idő, látványosan terhelné a böngészőt.

Mező	Leírás
id	a rekord egyedi azonosítója
~~safe~~	~~„biztonságos-e” a betűtípus~~
title	betűtípus neve
basic	szerepel az első teszt alap betűtípuslistájában
univ	szerepel az univerzális betűtípuslistában
feature	szerepel az újraválogatott listában
created	a később hozzáadott fontok beszúrásának időpontja, vagy „0000-00-00 00:00:00” eredeti elem esetén

9. táblázat - A *fingerprint_fonts* tábla új struktúrája

A származtatott betűtípuslisták előállításának algoritmusa:

1. `fonts_all` a JS detektálással kapott betűtípuslista,

   `fonts`, `fonts_univ` és `fonts_feature` kezdetben üres listák,

   `fonts_db` az eredeti betűtípuslista a halmaz attribútumokkal.

2. A `fonts_db` listán végigiterálva:

   ha `fonts_db[i] ∈ basic`, akkor `fonts[i] = fonts_all[i]`,

   ha `fonts_db[i] ∈ univ`, akkor `fonts_univ[i] = fonts_all[i]`,

   ha `fonts_db[i] ∈ feature`, akkor `fonts_feature[i] = fonts_all[i]`.

A betűtípusok hatékony újraválogatásának érdekében (bővebben: 5.1.2.1) Flash módszerrel is készül egy lista, amelybe a fontok név szerint, vesszővel elválasztott formában szerepelnek. Ebből a listából minden egyes új font (ha van) bekerül a *fingerprint_fonts* táblába a létrehozás idejével együtt (*created*). Utolsó metódus a létrehozott ujjlenyomat bejegyzés és a használt betűtípusok párosítása, vagyis kapcsolatok létrehozása a *fingerprints_x_fonts* táblában.

5.1.3.2 Pluginlisták

Pluginlisták esetében az Internet Explorer a „leggyengébb láncszem”, nem rendelkezik a többi böngésző által alkalmazott `navigator.plugins` listával, tehát a közvetlen pluginlista lekérdezés nem valósulhat meg. Ehelyett egy ún. beszúrásos tesztelést kell alkalmazni, melynek során a lekérdezett pluginokhoz az oldalba beszúrva létrejön egy-egy objektum példány (hogyha telepítve és engedélyezve van a plugin), ezektől lehet lekérdezni a plugin jellemzőit. A feladat megoldására találtam egy nyílt forráskódú JS osztályt[63], amellyel a feladat böngészőfüggetlen módon megvalósítható.

[63] http://www.pinlady.net/PluginDetect/

A fentiek következtében az univerzális pluginlista az IE-ben detektálható pluginokon kívül mást nem tartalmazhat. Tesztelések során kiderült, hogy további megszorítások is szükségesek: az IE bizonyos esetekben eltérő Flash verziót detektált, a Chrome beépített pdf megjelenítővel rendelkezik (nem detektálható Adobe Reader), a Windows Media Player pedig csak IE-ben detektálható. Ezen ismeretek alapján alakítottam ki az egyes pluginlistákat, amit a 10. táblázat foglal össze.

Plugin	**univerzális** ***(plugins_univ)***	**nem univerzális** ***(plugins_nonu)***	**összes** ***(plugins_all)***
Silverlight	✓	×	✓
VLC Player	✓	×	✓
Java	✓	×	✓
Shockwave Player	✓	×	✓
QuickTime Player	✓	×	✓
Flash Player	×	✓	✓
Adobe Reader	×	✓	✓
Windows Media Player	×	×	✓

10. táblázat - A kialakított pluginlisták

Az univerzális pluginok külön táblában (*plugins*) is tárolásra kerülnek, legfőképp statisztikakészítés céljából. A *plugins* tábla mezői: név, verzió és az ujjlenyomat rekord azonosítója.

5.1.3.3 Navigator hash

A window.navigator[64] objektum a böngésző és rendszer tulajdonságait írja le, nagyon összetett módon. Tartalmaz egyszerű értékeket (pl. *platform*), referenciákat más objektumokra (pl. *plugins*), konstans és böngészőfüggő attribútumokat egyaránt. Véleményem szerint önmagában is kiemelkedően jól azonosítja az egyes böngészőpéldányokat, ezáltal lokális azonosítóként működhet (pl. azonos IP cím tartományon belül). Feltételezésem igazolása érdekében a *fingerprints* tábla *navigator_hash* mezőjében tárolásra kerül az objektum sorosított, majd hash-elt változata. A sorosításhoz készítettem egy rekurzív JS függvényt, amely listák és objektumok (fa struktúrájú) ábrázolására is alkalmas. A hashelés szintén kliens oldalon történik egy nyílt forráskódú függvénykönyvtár[65] segítségével, hiszen feltételezésem igazolásához nincsen szükség a teljes objektumra, valamint a visszaküldött adat méretét

[64] https://developer.mozilla.org/en/DOM/window.navigator
[65] http://pajhome.org.uk/crypt/md5/

is sokkal kisebb mértékben növeli a hash változat (egyébként a sorosítással kapott string hossza meghaladja a teljes visszaküldött adat méretét).

5.1.3.4 Tracking ID és Passive ID

Létrehoztam egy egyszerű szerveroldali algoritmust, amely a rekordok előzetes csoportosítását segíti elő azáltal, hogy az új rekordhoz megkeresi az attribútumok azonossága alapján legközelebbi rekord azonosítóját – ez a Tracking ID. A Passive ID ennek egy olyan változata, amely nem használja a megadott felhasználónevet (csak a passzív módszerre támaszkodik), ezáltal követhető a felhasználónév változása.

Az algoritmus egy következtetési rendszer szerint működik a 11. táblázat alapján. A cellákban álló jelek a beszúrandó rekord és egy régebbi rekord attribútumai közötti relációkat mutatják.

Attribútum	Azonos böngésző	Új böngésző	Új hálózat	Új felbontás vagy időzóna
user_id_short	=	=	=	≠
ip	=	=	≠	=
user_agent	=	≠	= (a)	= (a)
plugins_univ	= (a)	=	=	=
fonts_feature	-	-	-	=
fonts_univ	= (a)	x	-	x
fonts_all	x	x	= (a)	x
user_name	= (b)	= (b)	= (b)	= (b)

11. táblázat - Tracking ID következtetési rendszer
(oszlopon belül azonos betű = ÉS kapcsolat, különböző betű = VAGY kapcsolat)

Akkor kaphat értéket a Tracking ID vagy Passive ID, ha a beszúrandó rekordhoz létezik az adatbázisban a relációs feltételeknek megfelelő rekord. Az „=” és „≠” alapfeltételek, együttes teljesülése esetén értéket kap a Passive ID, hogyha nincs találat, ami több feltételt is teljesít. Az azonos betűvel jelölt feltételeknek egyszerre kell teljesülnie (ÉS kapcsolat), de betűnként egymástól függetlenek (VAGY kapcsolat). Az „x” esetén tetszőleges, „-” esetén pedig nem értelmezett a reláció (mert egy másik miatt egyértelmű). A relációk megadásánál nehéz megtalálni azt az egyensúlyt, amivel biztosítható a megfelelő hibatűrés, és egyben a téves csoportosítások száma is minimalizálható.

5.1.3.5 FireGloves

A *fire_gloves* attribútum a FireGloves védekezési eszköz (ld. 6. fejezet) használati állapotát jelzi. A különböző értékek segítségével szét lehet választani a kiegészítő használata mellett, illetve a normál módban létrehozott bejegyzéseket, ami az eredmények kiértékelésekor fontos, hiszen a FireGloves jelentős mennyiségű „értéktelen" ujjlenyomatot tehet az adatbázisba. Négy különböző értéke lehet: „before" a FireGloves publikussá tétele előtti időszakot jelölve, „nodetect" a plugin megjelenése és a detektáló algoritmus beépítése közötti időszakban, „off" ha ki van kapcsolva (vagy nincs telepítve), és „on" ha be van kapcsolva (a mező értékeit magyarázza, hogy már az éles adatgyűjtés közben került bevezetésre, utólagosan).

5.1.4 Az új adatgyűjtő oldal

Az új weboldal tervezése során először meghatároztam a legfontosabb arculati, megjelenítési szempontokat. Legyen *nemzetközi* (többnyelvű), *felhasználó közeli* (érthető kezelőfelület, kapcsolat felvételi lehetőség), *figyelemfelkeltő* (teszt kipróbálása több böngészőben), *informatív* (a látogatók széleskörű tájékoztatása, GYIK), és nem utolsó sorban *privát szféra barát* (a teszt nem kezdődik el automatikusan). Ennek megfelelően a technikai kihívások a következők voltak: *rovatszerű tartalmak* kezelése, *menüpontok* kezelése, *moduláris bővíthetőség* (pl. kapcsolat modulban üzenetküldés), *nyelvi címkék* kezelése. Mindehhez szükség volt egy *adminisztrációs felület*re, a későbbi módosítások és bővítések is egyszerűbbek általa. A kialakított menüpontokat a 12. táblázat foglalja össze, a weboldal kezdőlapját a 9. ábra prezentálja. (További képernyőképek a weboldal egyes részeiről a mellékletben.)

Menüpont	Leírás
Ujjlenyomat teszt	Az ujjlenyomat teszt kipróbálása. A kísérlet leírása, tájékoztatás a védekezési módszerekről, tárolt adatok és adatvédelmi elvek ismertetése.
Cikkek	A témával kapcsolatos publikációink és elérhetőségük.
GYIK	A látogatók által gyakran feltett kérdésekre adott válaszok. Többnyire a kutatáshoz, az ujjlenyomat technikához, és a védekezési módszerekhez kapcsolódnak.
Partnerek	A kutatást segítő partnerek és támogatók listája.
Kapcsolat	Üzenetküldő felület.
FireGloves	A FireGloves plugin oldala. Letöltés, verzió napló, és felhasználói útmutató.

12. táblázat - Az egyes menüpontok tartalma

9. ábra - A Böngészőfüggetlen ujjlenyomat teszt 2.0 kezdőlapja

5.1.4.1 Technikai megvalósítás

A futtató környezet egy tetszőleges operációs rendszeren üzemeltetett Apache szerver, MySQL és PHP támogatással. A webes alkalmazást MVC (Model View Controller) tervezési minta alapján készítettem, a megjelenítéshez a Smarty[66] stíluslap kezelő rendszert használtam. A formázást W3C szabvány szerinti HTML5 és CSS3 leírónyelveken készítettem. Speciális feladatokra az alábbi osztálykönyvtárakat választottam: ADOdb[67] (adatbázis-kezelés), phpMailer[68] (szabványos e-mail), Securimage[69] (CAPTCHA ellenőrző). A kliens oldali fejlesztés hatékonyabbá tételéhez a jQuery[70] böngészőfüggetlen JavaScript API-t használtam, az adminisztrációs felületen a nicEdit[71] szövegszerkesztő segíti a HTML tartalom formázását. Az alkalmazás 6 fő

[66] http://www.smarty.net/
[67] http://phplens.com/lens/adodb/docs-adodb.htm
[68] http://phpmailer.worxware.com/
[69] http://www.phpcaptcha.org/
[70] http://jquery.com/
[71] http://nicedit.com/

modulból áll (lásd 13. táblázat), azonban ezek közül csak az ujjlenyomatozó módszert megvalósító *fingerprint* modul releváns, a többit érintőlegesen tárgyalom.

Modul	**Feladat**
content	a rovatszerű tartalmak kezelése
fingerprint	az ujjlenyomatozó technikát megvalósító modul
langtext	nyelvi címkék kezelése
login	adminisztrátori bejelentkezés
menu	menüpontok kezelése
message	üzenetküldés az adminisztrátoroknak

13. táblázat - A webes alkalmazás fő moduljai

A rovatszerű tartalmak jellemzője, hogy három részből állnak: cím, bevezető és tartalom. Listázáskor csak a bevezető jelenik meg (vagy a tartalom, ha nincs bevezető), és a „Bővebben »” gombra kattintva tekinthető meg a teljes tartalom – ez által nem kell oldalakat görgetni egy téma átugrásához. A szerkesztői felületen új tartalom hozzáadásakor a fentieken kívül meg kell adni a bejegyzés *nyelv*ét, és a *rovat*át is (melyik menüpontnál jelenik meg).

A nyelvi címkék a weboldal *megjelenítési* részéhez tartozó, nem tartalmi jellegű szövegek, melyekre a PHP és JavaScript forráskódokban egyedi nevükkel hivatkozunk, pl. az oldal címét *pageTitle* nevű címke tartalmazza. Új címke hozzáadásakor meg kell adni a címke *nevét*, a kapcsolódó *modult*, és a megjelenítendő *szöveget* nyelvek szerint (jelen esetben angolul és magyarul).

Az adminisztrátori beléptetést megvalósító *login* modul nem érhető el publikus linken keresztül (tehát ismerni kell hozzá a linket vagy a rendszert), illetve el van látva CAPTCHA ellenőrzéssel, a „nyers erővel” történő próbálkozások ellen.

Az egyes menüpontok a különböző rovatok és modulok elérését teszik lehetővé a *menu* modulon keresztül. Egy menüelem önmagában azonosíthat egy rovatot, tartalom bejegyzések rendelhetők hozzá, de kiválasztása egy modul betöltését is eredményezheti. Új menüpont hozzáadásakor meg kell adni a *menüpont nevét* (minden nyelven), és a menüben elfoglalt *helyét* (sorszám). Opcionálisan megadható *modulnév*, ha nincs megadva, akkor rovatként viselkedik, a hozzárendelt cikkeket tölti be. Rovat esetén kiválasztható a megjelenítés (cikkszerű vagy GYIK jellegű). Megadható továbbá a *class* attribútum értéke, amellyel egy eltérő stílus osztályt állíthatunk be.

Az üzenetküldő modulnak nincs adminisztrátori felülete, az előzetesen megadott e-mail címekre a rendszer elküldi az üzenetet. Az üzenetküldés a beléptetéshez hasonlóan CAPTCHA ellenőrzéssel működik a támadások elkerülése végett.

5.1.4.2 A fingerprint modul

Az ujjlenyomatmódszert a *fingerprint* modulban implementáltam. A modul teljes egészében AJAX módon működik, betöltése akkor indul, amikor a felhasználó rákattint az „Ujjlenyomat teszt indítása!" gombra. A betöltést a *fingerprintloader.js* fájlban valósítottam meg, ami ezen kívül tartalmazza a folyamatjelző csík (progressbar) vezérlőit, az Evercookie eltávolító eljárást, és a hibajelentés küldő funkciót is.

Betöltéskor a *fingerprint* modul a detektálandó betűtípuslistát (ld. 5.1.3.1 fejezet), a generált tesztazonosítót[72], és az ujjlenyomatozó scriptet elküldi a böngészőnek. A kliens oldalon futó program a kapott kódot beszúrja a dokumentum törzsébe, ezáltal elkezdődik az algoritmus futása.

A tesztet több részfázisra bontottam (14. táblázat), ezáltal működése jobban követhető, az állapotjelző csík szabályozása és a lépések megjelenítése is ez alapján történik. A script először a közvetlenül elérhető adatokat kérdezi le: nyelvi beállítás, használt operációs rendszer neve, képernyő tulajdonságok, időzóna, build ID, DNT,[73] és FireGloves használat. Ezt követik az összetettebb mechanizmusok: JS és Flash font detektálás (ld. 2.1.7.3, és 5.1.3.1 fejezet), pluginlisták beszúrásos lekérdezése (ld. 5.1.3.2 fejezet), támogatott formátumok és hozzá tartozó pluginok lekérdezése (mimeTypes), majd a *navigator hash* (ld. 5.1.3.3 fejezet) elkészítése. Az adatok küldését megelőző lépés az Evercookie segítségével eltárolt becenév kiolvasása, vagy ha nem áll rendelkezésre, akkor új becenév bekérése felugró ablakban (ld. 5.1.2.3 fejezet). Az adatok a detektálási folyamat alatt egy gyűjtőobjektumban vannak tárolva, amit a folyamat végén a script sorosított JSON formátumban AJAX POST metódussal elküld a szervernek. Az adatok szerveroldali feldolgozása után a kliens megkapja az eredményt, és egy felugró ablakban formázottan megjeleníti.

[72] Védelmi megoldásként a modul létrehoz egy tesztazonosítót (a *test_ids* táblában), amely a teszt indításától számítva 3 percig érvényes, és az adatok beérkezésekor törlődik. Érvényes azonosító hiányában a szerver figyelmen kívül hagyja a kérést, ezáltal elkerülve a hamis adatokkal való elárasztást. A módszer nem kijátszhatatlan, viszont az egyszerű csalásokkal szemben megfelelő védelmet nyújt.

[73] A *build ID* és *DNT* a fejlesztés idején csak Firefox böngészőkből volt elérhető JS API-ból.

	Fázis	Attribútumok
1.	Betöltés	

2.	Közvetlenül elérhető attribútumok lekérdezése	`locality`
		`os`
		`screen`
		`screen_avail`
		`timezone`
		`build_id`
		`dnt`
		`fire_gloves`

3.	Betűtípus detektálás	`fonts_all` `fontlist`[74]

4.	Plugin detektálás, támogatott formátumok, navigator hash	`plugins_univ`
		`plugins_nonu`
		`plugins_all`
		`mimetypes`
		`navigator_hash`

5.	Felhasználónév lekérdezése	`user_name`
6.	Adatok küldése	
Szerveroldali feldolgozás		
7.	Eredmények megjelenítése	

14. táblázat - Az ujjlenyomatozás fázisai és a kapcsolódó attribútumok

A szerver oldali feldolgozás a műveletek alapján szintén több fázisra osztható. Először az egyszerűen meghatározható attribútumok kapnak értéket: létrehozás ideje, IP cím titkosítva, IP cím első két bájtja titkosítva, UAS, és HTTP Accept fejlécek. A komplexebb műveletek sorban a következők: a betűtípuslisták előállítása és új fontok mentése (ld. 5.1.3.1 fejezet), böngésző verzió kinyerése a UAS-ből[75], majd *tracking_id* és *passive_id* (ld. 5.1.3.4 fejezet) meghatározása. Az adatok mentése három részből áll: ujjlenyomat rekord mentése, ujjlenyomat-betűtípus kapcsolatok (ld. 5.1.3.1 fejezet) mentése, és univerzális pluginok (ld. 5.1.3.2 fejezet) mentése. A szerver az eredményekből készítet objektumot JSON formátumban elküldi a kliensnek.

A hagyományos eredmények (detektált adatok) mellett a *fingerprint* modul több új tájékoztató jellegű információt is közöl a felhasználóval. Hasonló rekord keresésekor (Tracking ID, ld. 5.1.3.4 fejezet) az illeszkedő eset alapján a rendszer előállítja az *analízis eredményét*. Készít egy *felhasználóbarát azonosítót* is, amit úgy állít elő, hogy a generált MD5 azonosítót binárissá alakítja, és a bitek egy-egy csoportjának megfeleltet egy listából sorszám alapján kiválasztott elemet, ahol a sorszám a bitcsoport tízes számrendszerbeli értéke (15. táblázat). A detektált betűtípusok alapján

[74] A *fontlist* attribútum tartalmazza a Flash módszerrel detektált betűtípusokat vesszővel elválasztott formában (ld. 5.1.3.1).
[75] Az adatbázis bővítésekor bekerült új attribútum, melynek előállítását az előző tesztben létrehozott módszerrel (ld. 4.1.1) automatizáltam.

következtetni lehet a *számítógépre telepített szoftverekre*, a modul az alábbi négy szoftverre ad tippet: MS Office, MS Publisher, MS Outlook, és Adobe Creative Suite.

MD5 hash	32 hexadecimális számjegyből álló string							
bináris	128 bitből álló string							
bitcsoport	15 bit	16 bit	9 bit	7 bit	10 bit	1bit	10 bit	60 bit
lista	dátumok	időpontok	színek	gyümölcsök	nevek	ital	városok	kód

15. táblázat - Felhasználóbarát azonosító előállítása

	Fázis	**Attribútumok**
1.	Teszt validálása	

2.	Egyszerűen meghatározható attribútumok kiértékelése	`locality` (ha üres volt)
		`dnt` (ha üres volt)
		`created`
		`ip`
		`ip16`
		`user_agent`
		`accept_header`

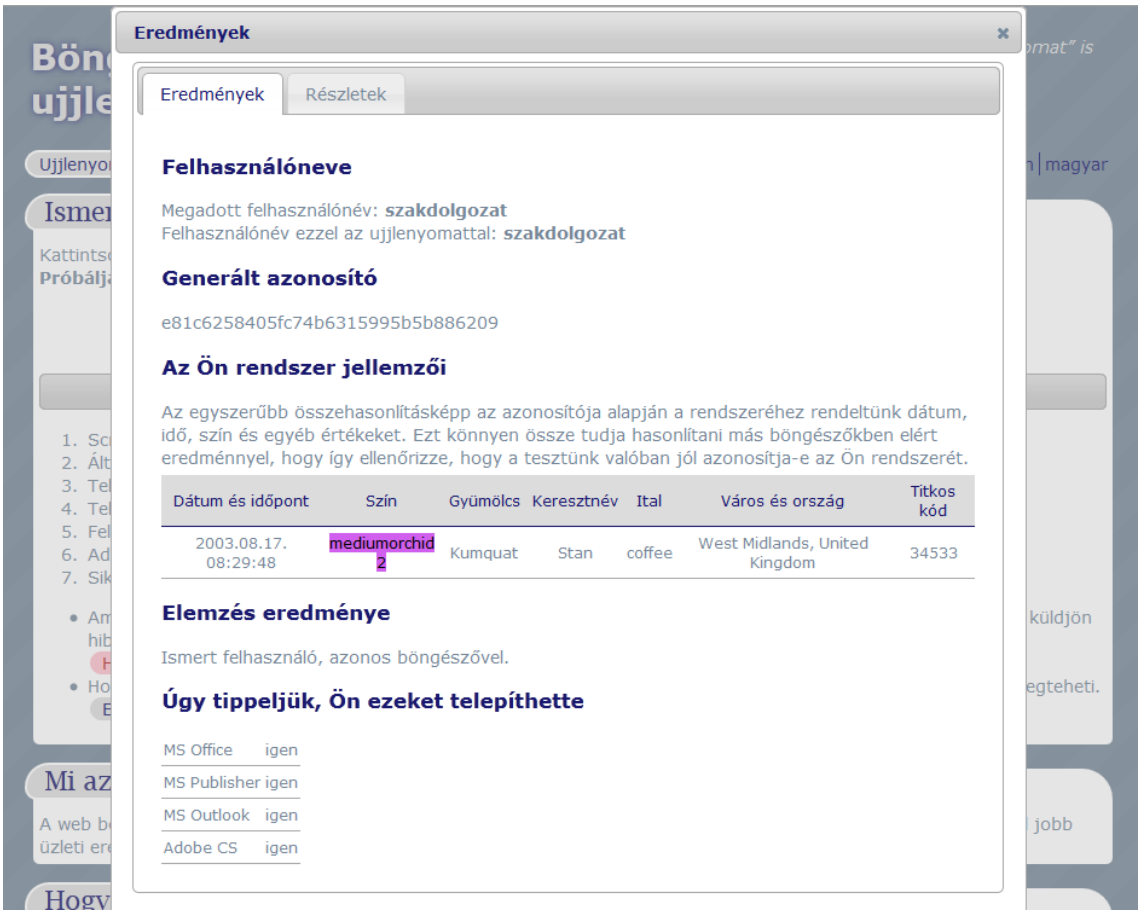
3.	Adatok validálása	

4.	Betűtípuslisták előállítása, használt betűtípusok száma, új fontok mentése	`fonts`
		`fonts_univ`
		`fonts_feature`
		`fonts_count`

5.	Böngészőverzió előállítása	`browser_ver`
6.	Tracking ID és Passive ID meghatározása	`tracking_id` `passive_id`
7.	Adatok mentése	
8.	Felhasználóbarát azonosító előállítása, szoftvercsomag tippek előállítása	
9.	Eredmények küldése a kliensnek	

16. táblázat - Szerver oldali feldolgozás fázisai és a kapcsolódó attribútumok

A visszaküldött eredmények által a felhasználó is sokat nyer, cserébe az adataiért. Kap egy felhasználóbarát azonosítót, amit könnyű összehasonlítani a különböző böngészőkben. Megtudja, hogy milyen rendszertulajdonságok kérdezhetők le böngészőjén keresztül, és ezek módosításával megpróbálhatja kijátszani az ujjlenyomatmódszert. Láthatja, ha a rendszer talált egy hasonló ujjlenyomatot, és az ehhez tartozó becenév alapján ellenőrizheti is, hogy valóban az övé-e. Ezen túl megismeri a rendszer beépített követő algoritmusának tippjét, és megtudja, hogy a rendszer mely szoftvereket ismerte fel a detektált betűtípusok alapján. Az eredmények megjelenítését a 10. ábra szemlélteti.



10. ábra - Eredmények megjelenítése a szerveroldali feldolgozás után

5.1.4.3 Javítások a külső scriptekben

A fontdetektáló algoritmus Linux rendszereknél hibát okozott, mivel ezek többsége esetén a böngésző alapértelmezett betűtípusa eltérő a detektáló algoritmusban összehasonlítási alapként használt „sans” betűtípustól. Ennek következtében a nem létező betűtípussal beszúrt elem dimenziója nem volt azonos az összehasonlítási alap dimenziójával, tehát telepítettnek lett detektálva. A megoldás módfelett egyszerű: „sans” helyett egy *nem létező betűtípussal* beszúrt elem lesz az összehasonlítási alap, ami a valódi alapértelmezett betűtípussal kerül beszúrásra, így tehát rendszertől és böngészőtől függetlenül is működőképes lesz a módszer.

Az Evercookie tároló eljáráson is módosítani kellett pár pontban. Az előzményekben tárolt azonosító a *history stealing* lehetőségének megszüntetése (ld. 2.1.7.1 fejezet) óta nem működőképes, ezért a funkciót kikapcsoltam. A pluginok betöltési kísérleteinek számát háromra csökkentettem, mert egyébként sok időt vesz igénybe. Ezen kívül kikapcsoltam az ETag, cache és Silverlight tárolási módokat, mert

többnyire nem működtek, az erőforrásigényük viszont nagy, jelentős módon lassították a futást.

5.1.5 Belső tesztelés

Az éles adatgyűjtés előtt szükséges volt a hibamentes működés ellenőrzése, hiszen több olyan változtatás volt, ami hibás működést okozhat bizonyos rendszerek esetén (pl. betűtípusok újraválogatása). A tesztelő csapat felkéréses úton került kialakításra, nemzetközi tesztelők is részt vettek a folyamatban. Több hasznos, építő jellegű visszajelzés érkezett, amelyek segítettek az oldal és a módszer jobbá tételében.

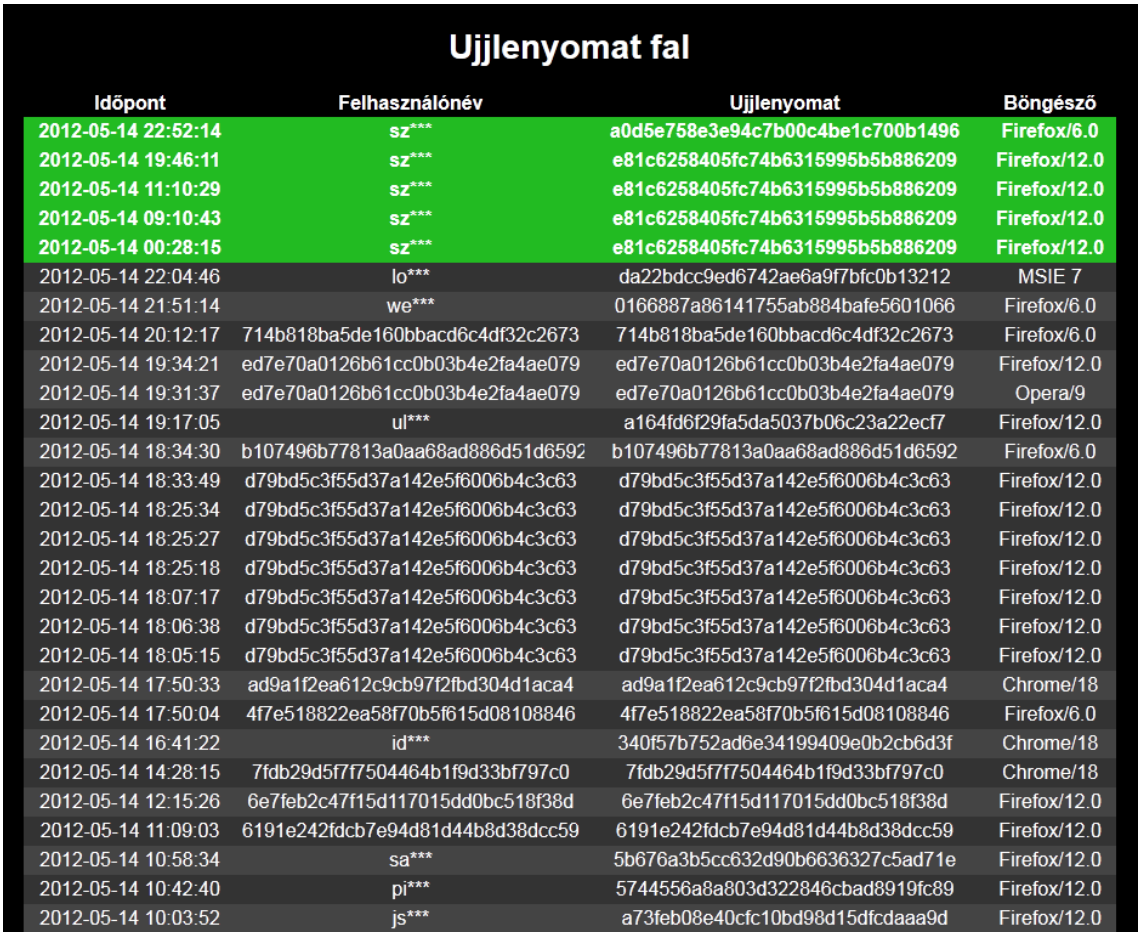
A teszt időszakban több mint 700 bejegyzés készült, ami elegendő adat az esetleges hibák javítására. Legfőbb cél a böngészőfüggetlenség biztosítása volt, kritikus pontja a válogatott betűtípuslista (*fonts_feature*), amely az azonosító domináns összetevője. A legtöbb eltérő detektálás Mac OS alatt történt Safari és Firefox böngészőkkel, a böngészőfüggő betűtípusok eltávolításra kerültek a listából. Bekerült a *hibajelentés funkció*, a HTTP fejléc alapú *automatikus nyelvválasztás* és az elárasztás ellen védelmet nyújtó *tesztazonosító* is, emellett az oldal tartalmi pontjaiban is történtek kisebb módosítások és kiegészítések. Fény derült arra is, hogy az Internet Explorer 6-os verziója nem rendelkezik JSON támogatással, ezt egy külső implementáció használatával oldottam meg. A hibák javítása által a rendszer fel lett készítve az éles adatgyűjtési folyamatra.

5.1.6 Sajtóesemény

A széleskörű adatgyűjtés érdekében sok látogatóra van szükség, ennek elérése a hírközlő médián keresztül a leghatékonyabb. Dr. Székely Iván és Gulyás Gábor György vezetésével előkészítettünk és lebonyolítottunk egy sajtótájékoztatót, amelynek a BME Híradástechnikai Tanszéke adott otthont és a BME-Infokom Innovátor Nonprofit Kft. támogatott, időpontja 2012. április 4-én volt. Több internetes hírportál képviseletében jelentek meg újságírók, készült rádiós interjú és egy tévés felvétel is. Az esemény nemzetközi kitekintéssel kezdődött az internetes privát szféra problémák kapcsán, amit a nyomkövetési technikák ismertetése követett, az ujjlenyomat módszerekre különös hangsúlyt fektetve. Ezután a tesztek bemutatása következett, először a Panopticlick projekt, majd a Böngészőfüggetlen ujjlenyomat teszt 2.0, több böngészővel és privát mód használatával is bemutattuk a módszer képességeit. Végül nyilvánosságra hoztuk a FireGloves kiegészítőt, és bemutattuk működését a két ujjlenyomat oldalon többféle

üzemmóddal. Több éles környezetben bizonyítottuk, hogy egyáltalán nem, vagy csak kis mértékben rontja a felhasználói élményt.

Az eseményre készítettem egy „Ujjlenyomat fal” nevű webes kisalkalmazást, amin az esemény alatt beérkező ujjlenyomatok voltak láthatók beérkezési sorrendben, felhasználónként csoportosítva (ld. 11. ábra). Táblázatos formában megjelent az ujjlenyomat készítésének időpontja, a megadott becenevek első két karaktere,[76] a kapott ujjlenyomat, valamint a böngészőprogram és verziója. Zöld háttér jelezte, hogyha valakinek sikerült azonos felhasználónév alatt különböző ujjlenyomatokat létrehozni.



Ujjlenyomat fal

Időpont	Felhasználónév	Ujjlenyomat	Böngésző
2012-05-14 22:52:14	sz***	a0d5e758e3e94c7b00c4be1c700b1496	Firefox/6.0
2012-05-14 19:46:11	sz***	e81c6258405fc74b6315995b5b886209	Firefox/12.0
2012-05-14 11:10:29	sz***	e81c6258405fc74b6315995b5b886209	Firefox/12.0
2012-05-14 09:10:43	sz***	e81c6258405fc74b6315995b5b886209	Firefox/12.0
2012-05-14 00:28:15	sz***	e81c6258405fc74b6315995b5b886209	Firefox/12.0
2012-05-14 22:04:46	lo***	da22bdcc9ed6742ae6a9f7bfc0b13212	MSIE 7
2012-05-14 21:51:14	we***	0166887a86141755ab884bafe5601066	Firefox/6.0
2012-05-14 20:12:17	714b818ba5de160bbacd6c4df32c2673	714b818ba5de160bbacd6c4df32c2673	Firefox/6.0
2012-05-14 19:34:21	ed7e70a0126b61cc0b03b4e2fa4ae079	ed7e70a0126b61cc0b03b4e2fa4ae079	Firefox/12.0
2012-05-14 19:31:37	ed7e70a0126b61cc0b03b4e2fa4ae079	ed7e70a0126b61cc0b03b4e2fa4ae079	Opera/9
2012-05-14 19:17:05	ul***	a164fd6f29fa5da5037b06c23a22ecf7	Firefox/12.0
2012-05-14 18:34:30	b107496b77813a0aa68ad886d51d6592	b107496b77813a0aa68ad886d51d6592	Firefox/6.0
2012-05-14 18:33:49	d79bd5c3f55d37a142e5f6006b4c3c63	d79bd5c3f55d37a142e5f6006b4c3c63	Firefox/12.0
2012-05-14 18:25:34	d79bd5c3f55d37a142e5f6006b4c3c63	d79bd5c3f55d37a142e5f6006b4c3c63	Firefox/12.0
2012-05-14 18:25:27	d79bd5c3f55d37a142e5f6006b4c3c63	d79bd5c3f55d37a142e5f6006b4c3c63	Firefox/12.0
2012-05-14 18:25:18	d79bd5c3f55d37a142e5f6006b4c3c63	d79bd5c3f55d37a142e5f6006b4c3c63	Firefox/12.0
2012-05-14 18:07:17	d79bd5c3f55d37a142e5f6006b4c3c63	d79bd5c3f55d37a142e5f6006b4c3c63	Firefox/12.0
2012-05-14 18:06:38	d79bd5c3f55d37a142e5f6006b4c3c63	d79bd5c3f55d37a142e5f6006b4c3c63	Firefox/12.0
2012-05-14 18:05:15	d79bd5c3f55d37a142e5f6006b4c3c63	d79bd5c3f55d37a142e5f6006b4c3c63	Firefox/12.0
2012-05-14 17:50:33	ad9a1f2ea612c9cb97f2fbd304d1aca4	ad9a1f2ea612c9cb97f2fbd304d1aca4	Chrome/18
2012-05-14 17:50:04	4f7e518822ea58f70b5f615d08108846	4f7e518822ea58f70b5f615d08108846	Firefox/6.0
2012-05-14 16:41:22	id***	340f57b752ad6e34199409e0b2cb6d3f	Chrome/18
2012-05-14 14:28:15	7fdb29d5f7f7504464b1f9d33bf797c0	7fdb29d5f7f7504464b1f9d33bf797c0	Chrome/18
2012-05-14 12:15:26	6e7feb2c47f15d117015dd0bc518f38d	6e7feb2c47f15d117015dd0bc518f38d	Firefox/12.0
2012-05-14 11:09:03	6191e242fdcb7e94d81d44b8d38dcc59	6191e242fdcb7e94d81d44b8d38dcc59	Firefox/12.0
2012-05-14 10:58:34	sa***	5b676a3b5cc632d90b6636327c5ad71e	Firefox/12.0
2012-05-14 10:42:40	pi***	5744556a8a803d322846cbad8919fc89	Firefox/12.0
2012-05-14 10:03:52	js***	a73feb08e40cfc10bd98d15dfcdaaa9d	Firefox/12.0

11. ábra - Az ujjlenyomat fal működés közben

Az esemény után több releváns hírportál[77] is beszámolt a kutatásról, cikkeikben mellékelve az ujjlenyomat oldal címét. Ennek hatására – a terv szerint – elindult a széleskörű adatgyűjtés, e mondat írásakor több mint 10 ezer bejegyzés szerepelt az ujjlenyomat táblában.

[76] Üres becenév esetén az ujjlenyomat került a helyére.
[77] Bővebb információk: http://pet-portal.eu/press/#ujjlenyomat2

5.1.7 Előzetes statisztika

A több böngészős tesztek aránya a várakozásokkal ellentétben hasonló lett, mint az első teszt esetén, a nagy számok törvénye miatt azonban szerencsésebb a helyzet. Több mint 1300 látogató legalább két böngészővel futtatta a tesztet (átlagosan 2,3).

A 17. táblázatban különböző bontások szerint kiszámítottam az egyes attribútumok bináris entrópiáját, ez segíti a vizsgálati halmazok összehasonlítását. Az előző teszthez képest egy bináris nagyságrenddel javult a *User ID* entrópiája, ami igazolja, hogy sikerült továbbfejleszteni a válogatott betűtípuslistát (ld. 5.1.3.1 fejezet). A UAS viszont ugyanennyivel csökkent, ez igazolja a feltételezés, miszerint a megváltozott összetétel kevésbé azonosító jellegű – a verziószámokon azonban még lehetne rövidíteni. Az egyedi betűtípuslisták számának növekedésében közrejátszik a JS fontdetektáló algoritmus javítása (ld. 5.1.4.3 fejezet), a bejegyzések között nem fordult elő hibásan detektált fontlista. Meglepő, hogy a limitált számú pluginokból álló *Plugins All* hasonló fokú egyediséggel bír, mint a rendkívül sok adatot tartalmazó MimeTypes (ld. 5.1.3.2 fejezet). Az univerzális pluginlistának jóval kisebb az azonosító képessége, ezt a böngészőfüggőség miatti megrövidülés okozza. Feltételezésem a *Navigator* objektummal kapcsolatban (ld. 5.1.3.3 fejezet) beigazolódott, miszerint az összes többi adatnál jobban képes azonosítani a böngésző példányt.

Attribútum	1 tesztet végzők (~4800)	Többször tesztelők[78] (~3200)	Trükk nélkül (~8000)	Összesen (~10000)
User ID	9,4	8,4	9,9	10,0
User Agent	6,3	6,6	7,3	6,9
Fonts All	10,0	9,6	10,3	10,8
Plugins Univ	7,5	7,5	8,4	7,8
Plugins All	10,1	9,5	10,4	10,8
MimeTypes	10,6	9,1	10,0	11,1
Navigator Hash	11,0	9,9	10,5	12,0
User Name	10,8	9,7	10,8	11,9
Fonts Real	9,8	9,4	10,1	10,6

17. táblázat - Attribútumok bináris entrópiája az ujjlenyomat táblában, többféle bontásban

Technikai adatok felmérésére az egyszer tesztelők halmaza a legideálisabb, hiszen minden bejegyzés biztosan más gépről származik. Képernyőfelbontást illetően átvette a vezetést az „1366x768” pixeles méret (16%), második helyen továbbra is az

[78] A felbontást, időzónát, operációs rendszert módosítók nem kerültek bele, csak azok, akik több böngészővel futtatták a tesztet.

„1280x1024” szerepel, előbbi notebook kijelzőkre, utóbbi asztali monitorokra jellemző. A látogatók 87%-a Windows, 5,3%-a Mac, 7,1%-a Linux vagy egyéb Unix rendszert használt, a maradék 0,6% ismeretlen. Legtöbben Firefox/11.0 böngészővel teszteltek (34,8%), második a Chrome/18 (27,5%), az IE 8-as vagy 9-es változatát 10%, az Opera 9-es verzióját 6,5%, a Safari valamely új verzióját 4% használta. Jelentős volt emellett a régebbi Firefox és Chrome verziók száma, összesen több mint 20%. Általánosságban elmondható, hogy a felhasználók nagy része frissíti a böngészőjét, naprakész verziókat használnak. A látogatók több mint 76%-a magyar nyelvű böngészőt használt, közel 20% angolt (ebből 14,3% amerikai angolt), ez jól tükrözi a hazai és nemzetközi látogatók arányát.

A feldolgozás tervezett módszertana

Bebizonyosodott, hogy a teljes módszer hibamentes működése három összekapcsolódó feltételtől függ: az azonosító legyen böngészőfüggetlen, hatékonysága maximális, és a kliens oldali ujjlenyomatozó script minden platformon hibátlanul működjön. Ennek megfelelően a feldolgozást két fő részre bontottam, első az azonosítóval, második a kliens oldali programmal foglalkozik.

5.1.8 A generált azonosító hatékonyságának elemzése és javítása

A böngészőfüggetlenség a felhasználónév szerinti csoportosítással könnyen megállapítható. Készítettem erre egy speciális lekérdezést, és kiderült, hogy 27 felhasználó eltérő azonosítókat kapott több böngészővel való teszteléskor, az azonosító tehát *nem böngészőfüggetlen*. Azonban ez javítható a böngészőfüggő betűtípusok megkeresése és eltávolítása által.

Másik fontos kérdés, hogy egyedileg azonosít-e minden a valóságban különböző felhasználót, vagy csak bizonyos korlátok mellett működik, és ha igen, mik ezek a korlátok? Az előbbi lekérdezés módosításával megkaphatók azok az ujjlenyomatok, amik nem egy felhasználót, hanem felhasználó csoportokat azonosítanak, ezzel anonimitási halmazokat létrehozva. Az egy alkalommal tesztelők *fele* került kettő, vagy annál nagyobb méretű anonimitási halmazba, a melyek közül a legnagyobb 94 ujjlenyomatot tartalmaz. A jelenség részletes vizsgálata választ ad a kérdésre, hogy lehet-e még finomítani az azonosítón, vagy elértünk egy határt, amin túl már nem különböztethetők meg böngészőfüggetlen módon a felhasználók.

Minden azonosító újragenerálható az adatbázisban tárolt adatok alapján. A célnak megfelelően érdemes létrehozni egy algoritmust, amely a rekordok attribútumaiból és a hozzájuk tartozó betűtípus bejegyzésekből ismételten előállítja az azonosítót. Ennek segítségével az alábbi iteratív módon finomítható az azonosító:

1. Böngészőfüggő elemek kiszűrése és elhagyása
2. Böngészőfüggetlen elemek keresése, azonosító bővítése
3. Összes azonosító újragenerálása (egy új oszlopban)
4. Eredmény vizsgálata böngészőfüggetlenség és egyediség tekintetében
5. Hogyha az eredmény nem kielégítő, ismétlés az 1. lépéstől.

Az univerzális plugin lista hozzácsatolásával ideiglenesen javulna a helyzet, azonban hosszú távú követésre jelen formájában nem alkalmas, hiszen plugin verzió frissítésekor (ami relatíve gyakori esemény) az azonosító is megváltozna. Célravezetőbb a *fonts feature* halmaz bővítése, méghozzá olyan fontokkal, amelyek nem képezik az alap operációs rendszer részét, hanem attól független szoftvereken keresztül kerültek telepítésre. A fontok keresési tartománya leszűkíthető: operációs rendszer szerinti csoportosítással a minden felhasználónál előforduló fontokat ki kell zárni, pl. egy *default* halmaz (attribútum) létrehozásával a betűtípusok táblájában (*fingerprint_fonts*). A fontok böngészőfüggetlenségét bővítés előtt érdemes vizsgálni, mert kisebb munka, mint utólag kiszűrni a hibás elemeket. Ez a művelet algoritmikus úton elvégezhető, amennyiben a vizsgált font szerepelt a JS detektálási listában: hogyha OS-enként[79] elérhető volt a népszerű böngészők mindegyikéből, akkor böngészőfüggetlen.

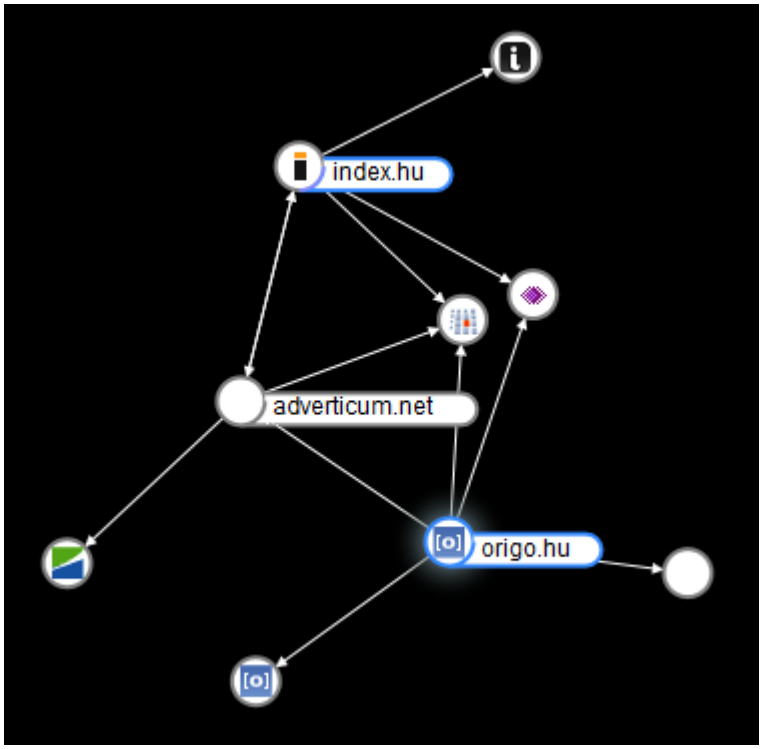
5.1.9 Optimalizálási lehetőségek a kliens oldali programon

A böngészőfüggő betűtípusok kiszűrése után a JS fontdetektáláshoz használt font adatbázis mérete is nagymértékben csökkenthető azáltal, hogy csak a böngészőfüggetlen fontok kerülnek bele. Ezzel jelentősen rövidíthető a detektálás futásideje – ez főleg a gyengébb konfigurációknál kritikus kérdés. A folyamat elrejtése érdekében csoportokra bontva is történhet a detektálás kis késleltetések beiktatásával, ezzel elkerülhető a sokkszerű terhelés.

Éles környezetben nagyon fontos az észrevétlen működés. Egy reklámokkal teli weboldal nagyságrendekkel lassabban jelenik meg, hiszen sok harmadik féltől – külső szolgáltatótól – származó tartalmat kell betöltenie a böngészőprogramnak (ld. 12. ábra).

[79] Linux és Unix alapú rendszereknél IE-vel és Safarival nem kell számolni.

Több külső plugint használó objektumot is tartalmazhat (pl. Flash), illetve a reklámok és widgetek nagy része JavaScript kód formájában kerül beágyazásra.



12. ábra - Objektumok betöltése külső szolgáltatók szervereiről a Collusion Firefox kiegészítő vizualizációjában, az index.hu és origo.hu hírportálok megtekintésekor

6 FireGloves

Az ujjlenyomat technikák viszonylag újak, nincs ellenünk publikus kidolgozott védekezési eszköz. Az eddig megismert ujjlenyomat módszerek többsége a 0 fejezetben ismertetett adatokat használja fel azonosító létrehozásához, ezen adatok mindegyike valamely rendszer vagy böngésző beállítás értékét reprezentálják. Ezek megváltoztatásával az ujjlenyomat eljárás megtéveszthető, azonban jellegükből adódóan (pl. képernyőfelbontás) módosításuk körülményes – ujjlenyomatváltáshoz pl. át kell állítani a tényleges képernyőfelbontást, vagy megváltoztatni a rendszer időzóna beállítását, de a biztos siker elérése érdekében plugin tiltása, vagy betűtípus törlése is szükséges. Ez láthatóan túl körülményes és túl nagy ár a követhetetlenség elérése érdekében, ráadásul laplekérésenként nem is elvégezhető.

A leghatékonyabb eszköz a JavaScript és a pluginok letiltása, ám ez elrontja a legtöbb weboldal funkcionalitását, használatukat ellehetetleníti. Szükség van egy megoldásra, amely a felhasználói élmény megtartása (vagy csekély mértékű romlása) mellett védelmet nyújt az új módszerekkel szemben. Természetesen a legjobb megoldás az volna, hogyha ez nem külső eszköz által valósulna meg, hanem a böngészőgyártók figyelmét sikerülne felhívni a probléma jelentőségére, és ők maguk beépítenék a szükséges védekezési mechanizmusokat programjaikba.

Helyzetkép: anonimitási halmazok

A JavaScript API-n keresztül elérhető attribútumok értékkészlete nagy, ezért sokféle kombinációban előfordulhatnak. Általánosságban elmondható az is, hogy egy paraméter megváltoztatása nem elegendő, amennyiben a többi változatlan marad, jó eséllyel követhető marad a felhasználó. Ennek következtében nehéz feladat a beállításoknak egy olyan kombinációját előállítani, amely által biztosítható a követhetetlenség.

A probléma többféleképpen megközelíthető. Az egyik lehetőség szerint azonos beállítások használatával kialakíthatók nagy anonimitási halmazok, amelyekben a felhasználók nehezen, vagy egyáltalán nem különböztethetők meg. Azonban ez csak sok felhasználó összefogásával lehetséges, ezen kívül az anonimitási halmazok szolgáltatónként, térségenként is változhatnak az egyes eszközök népszerűségének

függvényében. A módszer globális érvényű anonimitási halmazok kialakításával működne megfelelően, ám ennek kivitelezése nagy kihívás a változatos eszköz- és szoftverhasználat miatt (1680x1050 pixeles felbontás helyett zavaró lenne 1366x768 pixeles felbontás használata, csak a nyomkövetés kivédése érdekében).

Egy másik, inkább felhasználóbarát megközelítés a JavaScript API-n keresztül lekérdezhető attribútumok *korlátozása* vagy *meghamisítása*, ezáltal nem kell ténylegesen megváltoztatni a rendszerbeállításokat, a felhasználói élmény sokkal kisebb mértékben csorbul. Ehhez a módszerhez hasonlót használ pl. a JonDoFox anonim böngésző, amely több attribútumnál egységesen azonos értéket állít be, azonban vannak hátrányai: nem terjed ki mindenre, pl. a tényleges képernyőfelbontás továbbra is elérhető, és emellett túlzott szabályozásával sokat ront a felhasználói élményen.

Véleményem szerint az attribútumok gyakori (véletlenszerű) módosítása is jó megoldás lehet, hiszen az ujjlenyomatozó eljárást mindig új azonosító generálására készteti, és sokkal kevesebb felhasználó elegendő a rendszer megzavarásához. Nagyon jól működhet a módszer olyan rendszerek esetén, amelyek azonosító generáláshoz nem használnak IP címet (ilyen az általam fejlesztett módszer is, illetve vizsgálatom alapján úgy tűnik, hogy a BlueCava megoldása is).

A cél mindegyik megközelítésben közös: az adatok *értéktelenné tétele* által a szolgáltató legfeljebb az IP címre alapuló nyomkövetést legyen képes megvalósítani. Ezen tovább lehet javítani az IP cím elfedésével, pl. proxy szerveren, vagy MIX-eken keresztüli csatlakozással. A JonDoFox és a Tor Browser anonim böngészők tartalmaznak anonimizáló hálózatokhoz való csatlakozási lehetőséget külön alkalmazások segítségével, használatuk nagyon egyszerű, nem igényel különösebb szakmai ismereteket.

A kiterjesztés tervezése és implementálása

Az általam tervezett demonstrációs célú védekezési eszköz egy Firefox add-on (kiterjesztés), amely az ujjlenyomatozó technikák által használt attribútumokat hamisítja meg: képes azok tiltására, felhasználó általi szerkesztésére, és véletlenszerű értékek használatára is, beállítástól függően. Telepítéskor olyan alapértelmezett beállításokkal érkezik, amelyek jó eséllyel nagy anonimitási halmazba juttatják a felhasználót. Alkalmazásával a valódi rendszerbeállítások módosítása nélkül meghamisítható például az operációs rendszer típusa, képernyőfelbontás, időzóna. Mindemellett letilt bizonyos

Firefox specifikus objektumokat is, amik felhasználhatók ujjlenyomatozáshoz, vagy teljesítmény alapú teszteléshez (ilyenek például a `window.Components`, vagy a `window.performance.timing`). Az eszközt *privát böngészési mód*dal együttes használatra terveztem, hiszen aki el szeretné kerülni a nyomkövetést, a perzisztens tárolók rendszeres ürítését sem hagyhatja figyelmen kívül.

6.1.1 Működési elv

A működés alapelve az, hogy az ujjlenyomatozó módszerek által lekérdezett attribútumok értékét megváltoztatjuk azelőtt, hogy a böngésző feldolgozná a JavaScript kódot, ezáltal futáskor már a módosított értékeket olvassa ki a script. Ennek megvalósításához a közvetlenül letöltés után (feldolgozás és futtatás előtt) kell módosítani a forráskódot, majd visszaadni a böngészőprogramnak.

Az attribútumok értékének módosítására két megoldás jutott eszembe (ettől függetlenül lehet, hogy több is van). Első megoldás: az összes érintett objektum (vagy attribútum) megkeresése és kicserélése előre meghatározott „álobjektumra”, az összes letöltött JavaScript és HTML forráskódban (ld. 18. táblázat). Az álobjektumok ugyanazokat az attribútumokat tartalmazzák, mint a valódi, de *hamis* értékekkel. Ez hatékonyan megvalósítható például reguláris kifejezéses csere használatával, ám nagyméretű fájlok esetén komoly pluszmunkát jelenthet, lassíthatja az oldalak megjelenítését. Az attribútumok többféleképp is elérhetők[80], ezért alapos körültekintéssel kell kezelni őket, hogy ne maradjon ki egy lehetőség sem. Második megoldás: az érintett objektumok prototípusának attribútum lekérdezési függvényeinek felüldefiniálása. Ez a módosítás az adott oldal JavaScript névterén belül kiterjed az összes azonos típusú objektumra, tehát elegendő a HTML dokumentum legelső `<script>` eleme elé beszúrni a módosító kódot.

[80] Például `window.screen.width = window.screen['width'] = window['screen'].width = window['screen']['width'] = screen.width = screen['width']`, stb. Van olyan objektum, ami rendelkezik `getAttribute()` metódussal is.

Objektum és attribútum	Helyettesítés „álobjektummal”	Felüldefiniálás prototípusban
`window.screen.width`	`{ width: 800, height: 600 }.width`	`Window.prototype .__defineGetter__('screen', function() { return { width: 800, height: 600 }; }); [...] window.screen.width`
a képernyő valódi szélessége pixelben	800	800

18. táblázat - Az adatok meghamisításának kétféle módja

Egyszerűsége és kis erőforrásigénye miatt jelen implementációban a második megoldást választottam, azonban néhány speciális eset miatt a későbbiekben mindkét módszert alkalmazni fogom.

6.1.2 Módosított adatok

A FireGloves védelmi eszköz számos adatot képes letiltani vagy meghamisítani, utóbbi esetben lehet véletlenszerű (minden oldal lekérésnél változik), vagy a felhasználó által megadott értékű. A módosított adatok listáját a 19. táblázat szemlélteti, amelyben adatonként feltüntettem a megoldás módját, a kiegészítő által használt alapértelmezett értéket[81], valamint azt, hogy a felhasználó milyen formában képes szabályozni a beállítások ablakban.

Információforrás	Megoldás	Alapértelmezett	Beállítás
User Agent String	Célszerű gyakran előforduló értéket megadni, hogy ne legyen egyedi.	Mozilla/5.0 (Windows NT 6.1; rv:6.0) Gecko/20100101 Firefox/6.0	Szöveg
HTTP Accept Language		en-us, en	Szöveg
Span, Div és Paragraph elemek offsetWidth és offsetHeight attribútuma (JS font detektálás)	Konstans érték (pl. 0 esetén egy betűtípus sem detektálható) Véletlen érték (minden betűtípust telepítettnek detektál a módszer)	Aktív	Kapcsoló

[81] Az alapértelmezett értékeket a legnagyobb anonimitási halmaz szerint határoztam meg, a Panopticlick projekt weboldalán próbálgatásos teszteléssel (a kiegészítő használatával).

Információforrás	Megoldás	Alapértelmezett	Beállítás
A teljesítmény monitorozására szolgáló időpont jegyzék (a kliens gép teljesítménye is mérhető vele)	Konstans érték (pl. 0)	Aktív	-
Pluginok listája	Üres lista. A pluginok ettől függetlenül továbbra is működnek.	Aktív	Kapcsoló
Mime Type lista			
Képernyő felbontás	Gyakori érték, vagy listából véletlenszerűen választott.	1280x800	Szöveg
Operációs rendszer kódja		Win32	
Nyelvi beállítás kódja		en	
Időzóna beállítás		-1	Szám
Components objektum[82] (Firefox specifikus azonosítókat tartalmaz)	Üres lista.	Aktív	-
Cookie-k	Privát mód gondoskodik a törlésükről.	-	-
HTML5 tárolók			
Plugin tárolók			
Előzmények			

19. táblázat - A FireGloves által módosított adatok

Az IP cím elrejtését nem oldja meg, ahhoz proxy szerveren vagy anonimizáló hálózaton keresztüli csatlakozás szükséges. Emellett a pluginok által elérhető információforrások is kívül esnek hatáskörén[83], célszerű mellette plugin blokkoló kiterjesztés alkalmazása, mint például Adblock Plus, vagy NoScript – későbbi fejlesztésként elképzelhető, hogy kerül bele ilyen.

6.1.3 Fejlesztés

A szakmailag leginkább izgalmas kérdés a fejlesztés során az volt, hogy be lehet-e szúrni az attribútumok módosításához szükséges kódrészletet a letöltött tartalomba, mielőtt azt a böngésző feldolgozza, és ha igen, akkor hogyan? A választ sok kutatás után sem kaptam meg, ezért elkezdtem olyan kiterjesztések után nyomozni, amelyekről feltételeztem, hogy az említett mechanizmussal működnek. Hosszas keresés után rábukkantam a *Javascript Deminifier*[84] nevű *nyílt forráskódú* kiegészítőre, ami a tömör formájú „whitespace” karakterek nélküli JavaScript kódot visszaállítja tördelt,

[82] https://developer.mozilla.org/en/Components_object
[83] Ehhez a plugin forrását is módosítani kellene, azonban ez bonyolultabb feladat.
[84] https://addons.mozilla.org/hu/firefox/addon/javascript-deminifier/

olvasható formára[85]. Rövid tanulmányozás után konstatáltam, hogy a tervemnek megfelelően működik, ezért úgy döntöttem, hogy ez lesz munkám alapja.

A kapcsolódó programrész az alábbi módon működik: minden lekérésnél megvizsgálja, hogy a letöltendő fájl típusa a megadott dokumentum típusok közé tartozik-e, és amennyiben igen, akkor egy esemény által figyeli HTTP válasz befejeződését. A FireGloves esetében HTML formátumú dokumentumok a relevánsak és a HTML kódok első `<script>` eleme elé célszerű beszúrni a módosítást végrehajtó scriptet. Az esemény bekövetkezésekor rendelkezésre áll a letöltött dokumentum forráskódja, amit át kell adni a módosítást végrehajtó függvénynek. A függvény elvégzi a beszúrást vagy cserét, és a módosított forráskódot visszaadja a böngészőnek, amely ezt követően elvégzi annak feldolgozását ugyanúgy, mint normál esetben.

A *Javascript Deminifier* kiterjesztés tartalmazott megvalósítást az ablakonkénti működési állapot tárolására, hasznossága miatt megtartottam ezt a funkciót. A teljes értékű működéshez azonban tovább kellett fejleszteni, szükség volt speciális (Chrome API) eszközök alkalmazása is.

6.1.3.1 Kiterjesztés és böngésző preferenciák

A felhasználók tetszőlegesen módosíthatják az egyes attribútumok értékét (a 19. táblázat *beállítás* oszlopának megfelelően), és szabályozhatják a program működését (pl. véletlenszerű mód, indítási módok). Ezeket ún. *preferencia bejegyzésekben* kell tárolni, amelyeknek alapértelmezett érték is adható. A böngésző összes beállítása elérhető ilyen módon, használata a preferenciák kezelését megvalósító interfészeken[86] keresztül lehetséges.

A *User Agent String* és az *Accept Language* attribútumok módosítását nem csak a JavaScript API-ban kell elvégezni, hiszen ezeket HTTP fejlécben is elküldi a böngésző. A fejlécek értékeit a böngésző preferencia bejegyzésein[86] keresztül lehet módosítani, ennek módja az alábbi: a védelmi eszköz bekapcsolásakor, vagy a beállítások módosításakor átírjuk a megfelelő preferencia bejegyzések értékeit; kikapcsolásakor visszaállítjuk az eredeti értéket.

[85] A kiegészítő a JSBeautifier algoritmusát használja: http://jsbeautifier.org/, de erre a funkcióra nincs szükség, ezért eltávolítottam a kapcsolódó programrészeket.
[86] https://developer.mozilla.org/en/Code_snippets/Preferences

6.1.3.2 Privát böngészési mód detektálása

A kiterjesztés elindul a privát böngészési mód bekapcsolásakor, de működése aktiválható a kiegészítő sávon is, hogyha rákattintunk a „FireGloves” címkére. A privát mód detektálásához a gyártó által javasolt interfészt használtam[87]. Privát böngészési módban új fül nyitásakor automatikusan bekapcsolt állapotba kerül a program.

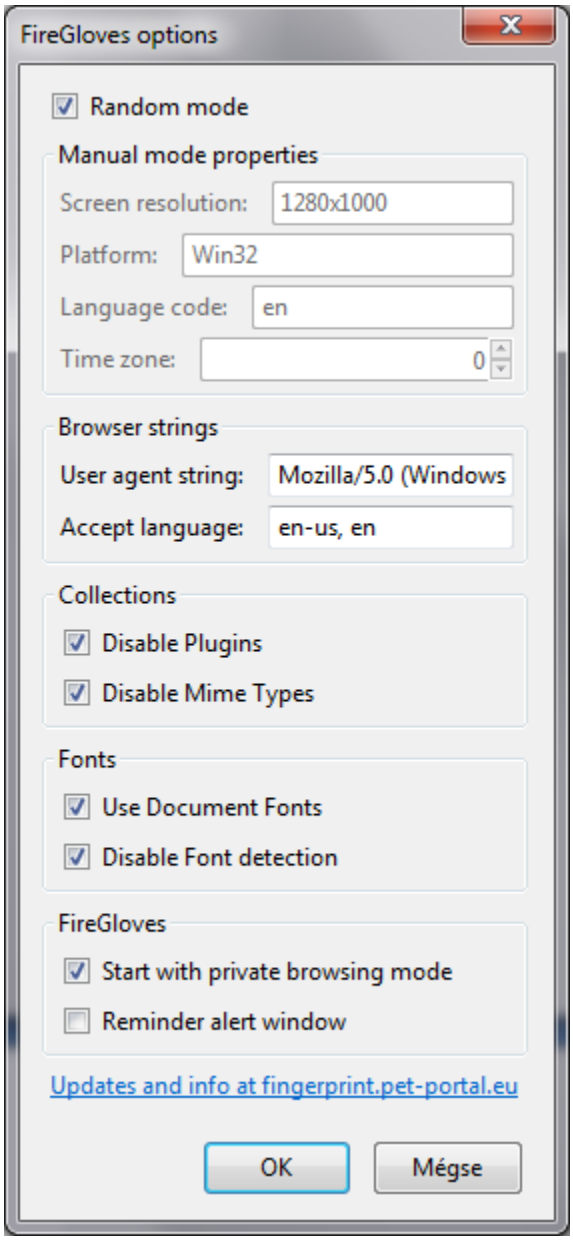
6.1.3.3 Beállítások párbeszédpanel

A beállításokat párbeszédpanelen[88] lehet módosítani (13. ábra), amit a kiegészítők listájánál a FireGloves kiterjesztés melletti „Beállítások” gombra kattintva lehet megnyitni. A beállítások az alábbiak (1.1-es verzió):

- *Random mode* (kapcsoló): véletlenszerű mód bekapcsolása. Használatakor a képernyő felbontás, operációs rendszer, nyelvi beállítás és az időzóna értékek előre elkészített listából kerülnek kiválasztásra véletlenszerűen.
- *Manual mode properties* (keret): A véletlenszerű mód kikapcsolt állapotában tetszőleges értékek megadására ad lehetőséget a keretben foglalt paraméterekre.
    - *Screen resolution* (szöveges mező): A képernyő felbontás értéke „WxH” alakban, ahol W a szélesség, H a magasság pixelben.
    - *Platform* (szöveges mező): Az operációs rendszert jelölő kód.
    - *Language code* (szöveges mező): A nyelvi beállítás kódja.
    - *Time zone* (szám mező): Időzóna eltolódás értéke (egész szám, negatív is lehet).
- *Browser strings* (keret): A HTTP fejlécekben küldött böngészőleíró paraméterek beállításai.
    - *User agent string* (szöveges mező): A böngésző leíró karakterlánc.
    - *Accept language* (szöveges mező): Tartalomra vonatkozó elfogadott nyelvek.
- *Collections* (keret): A lekérdezhető listákra vonatkozó beállítások.
    - *Disable Plugins* (pipadoboz): Plugin lista letiltása. Hogyha be van pipálva, akkor nem kérdezhető le a pluginok listája, viszont a pluginok ugyanúgy használhatók, nem lesznek letiltva. Hogyha nincs bepipálva, akkor a működése a szokásos, minden plugin lekérdezhető.
    - *Disable Mime Types* (pipadoboz): Hasonlóan a plugin listához.

[87] https://developer.mozilla.org/En/Supporting_private_browsing_mode
[88] https://developer.mozilla.org/en/Mozilla/Preferences/Preferences_system

- *Fonts* (keret): Betűtípusok használata, fontdetektálás tiltása
    - *Use Document Fonts* (pipadoboz): A beállításokon keresztül is elérhető opció, engedélyezése esetén használható tetszőleges betűtípus, egyébként csak néhány standard fontot enged.
    - *Disable Font detection* (pipadoboz): A betűtípus detektálásának tiltása. Hogyha be van pipálva, akkor az *offsetWidth* és *offsetHeight* értéke 0 (egy betűtípus sem detektálható). Hogyha nincs bepipálva, akkor az *offsetWidth* és *offsetHeight* értéke a valóshoz közeli véletlen érték (minden betűtípus telepítettnek lesz detektálva).
- *FireGloves* (keret): A kiegészítő egyéb beállításai
    - *Start with private browsing mode* (kapcsoló): A kiterjesztés állapota legyen aktív a privát böngészési mód indításakor.
    - *Reminder alert window* (kapcsoló): A kiterjesztés futására emlékeztető figyelmeztető ablak megjelenítése.



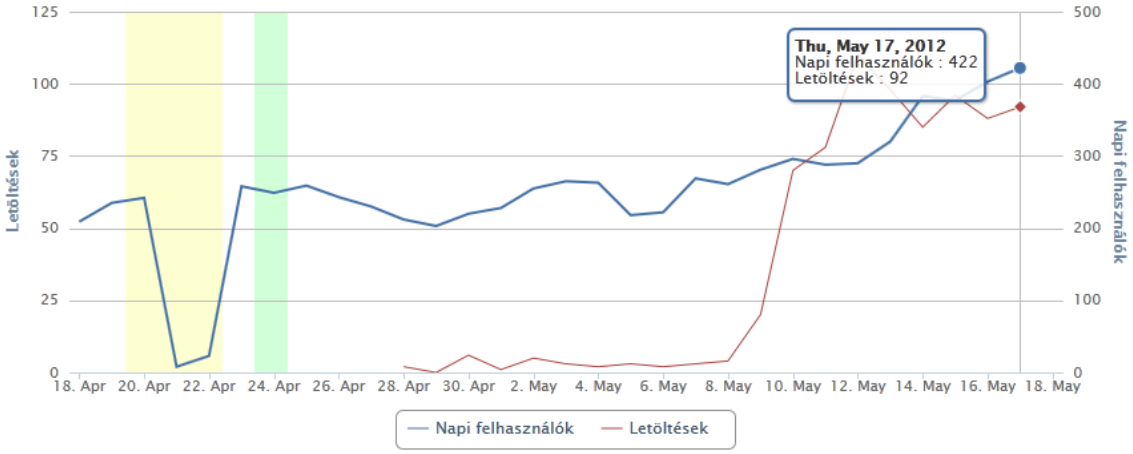
13. ábra - A FireGloves kiterjesztés beállításai

6.1.4 A FireGloves nyilvános bemutatása

A kiterjesztés első publikusan elérhető változatát pár nappal a sajtótájékoztató után jelentettem meg a *Böngészőfüggetlen ujjlenyomat teszt 2.0* weboldal *FireGloves* menüpontja alatt. Ugyanitt elhelyeztem egy rövid telepítési és felhasználói útmutatót, és egyéb információkat a szoftver működésével kapcsolatban.

Később regisztráltam a Mozilla *Kiegészítők (Add-ons)* weboldalán, ezzel egy időben már fejlesztettem a következő (1.1-es) verziót. Három hetes várakozás után elbírálták és jóváhagyták (már az újabb verziót), ezáltal a kiegészítők nyilvántartásában

is megjelenik, és bárki által elérhető. A felhasználók száma folyamatosan növekszik, jelenleg átlagosan naponta több mint 350-en használják (ld. 14. ábra).



14. ábra - *FireGloves* felhasználói és letöltési statisztika a Mozilla *Add-ons* oldaláról

A FireGloves elérhető a Mozilla Firefox Add-ons weboldalán az alábbi linken: https://addons.mozilla.org/hu/firefox/addon/firegloves/.

6.1.5 Együttműködés más PET technikákkal

A kiterjesztés akadályok nélkül használható *JonDoFox* és *Tor Browser* anonim böngészőkkel, és egyéb blokkolást segítő kiegészítőkkel, mint pl. *NoScript*, vagy *Adblock*. Tesztjeim alapján *JonDoFox*-szal kombinálva képes megváltoztatni a BlueCava algoritmusával generált azonosítót (véletlenszerű módot használva minden lekérésnél újat hoz létre), azonos beállításokkal pedig képes különböző rendszerekben megegyező azonosító előállítására.

6.1.6 Továbbfejlesztési lehetőségek

A jelenleg aktív 1.1-es verzió több fejlesztést is tartalmaz:

- Indítási beállítások: privát böngészési móddal való indítás kikapcsolható, az ekkor megjelenő emlékeztető ablak szintén kikapcsolható.
- Betűtípus beállítások: külön dobozba kerültek, közvetlenül állítható a dokumentum betűtípusainak használata.
- Értesítést kaptunk a módszert kijátszó lehetőségről[89] (Georg Koppen-től, a JonDoFox fejlesztőjétől), az új verzióban már nem alkalmazható.

[89] A módszert kijátszó lehetőség: http://fingerprint.pet-portal.eu/files/FGUnhook.html

Több építő jellegű visszajelzés érkezett már a kiterjesztéssel kapcsolatban a működésre és beállításokra vonatkozólag, ezeket figyelembe veszem a jövőbeni fejlesztések során.

A védelmi eszköz normál módban történő használatakor az aktiválás előtt megnyitott oldalon a módosítatlan kód szerepel, ezért a védelem csak annak újratöltésekor lép életbe, célszerű tehát a bekapcsolást követően automatikusan újratölteni az adott oldalt. Szükséges további attribútumok felvétele a beállításokhoz, mert néhány közülük kimaradt (pl. *navigator.appVersion*), és ezekből továbbra is kiolvasható a valódi információ.

Georg Koppen több megkerülési módról is küldött visszajelzést, amelyből két esetben új *Window* példány – és ezzel együtt új JavaScript névtér – létrehozásával elérte az attribútumok valódi értékét. Ez azért lehetséges, mert ezekbe nem tölt be külső tartalmat, ezzel megkerülve az algoritmust, amely pont ezt figyeli. Az egyik lehetőség `<iframe>` elemen keresztül a *contentWindow* objektum használata, másik pedig az üres ablak nyitása (`window.open("about:blank", "")`) által létrehozott új `window` objektumon keresztül lehetséges. Az első lehetőség javítva lett az 1.1-es verzióban (attribútum felüldefiniálással), a második azonban bonyolultabb, mert az új névtér miatt a `window` objektum az eredeti prototípussal jön létre.

A jelenlegi (1.1-es) verzió az `offsetWidth` és `offsetHeight` attribútumok felüldefiniálásával hatástalanítja a JS fontdetektáló módszert, azonban ez bizonyos weboldalak speciális vezérlőinek (pl. jQuery UI elemek) működésében kárt okoz a hamis értékek miatt. A megoldást azonban megtaláltam: a `style` CSS leíró objektum prototípusában kell felüldefiniálni a `fontFamily` attribútum `set` metódusát valamint a `setAttribute("font-family", "[betűtípus]", [prioritás])` függvényt. Ennek módja nem egyértelmű, hogyha funkcionalitásukat szeretnénk megtartani, pl. limitált mennyiségű, vagy adott listában szereplő fontokat engedélyezni szeretnénk. Emellett CSS tartalomszűrésre is szükség van, hiszen azon keresztül is érkezhetnek betűtípus beállítások. A fejlesztés folyamatban van, a következő verzió tartalmazni fog korlátozást a használt betűtípusokra, ennek alapja egy előre megadott fontokat tartalmazó lista, amit a felhasználó módosíthat.

7 Irodalomjegyzék

[1] G. Gy. Gulyás, R. Schulcz and S. Imre, „Comprehensive Analysis of Web Privacy and Anonymous Web Browsers: Are Next Generation Services Based on Collaborative Filtering?," *Proceedings of the Joint SPACE (Sustaining Privacy in Autonomous Collaborative Environments) and TIME (Trust in Mobile Environments) Workshops*, pp. 17-31, 2008.

[2] Gulyás G. Gy., *Felhasználók nyomkövetése a weben, és kapcsolódó védekezési módszerek.* [Előadás]. 2011. 07. 08..

[3] A. Loveless, "EU Cookie Law – it ain't all about cookies (or browsers)," enchiladadigital.com, 2011. 07. 24. [Online]. http://www.enchiladadigital.com/2011/07/24/eu-cookie-law-cookies-browsers/. [2012. 05. 20.].

[4] Gulyás G. Gy., „Anonim-e az anonim böngésző?," *Alma Mater 6. évfolyam, 1. szám.*

[5] T. Narten and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," IETF.org, 2001. január [Online]. http://www.ietf.org/rfc/rfc3041.txt. [2012. 05. 20.].

[6] Hullám G., „A web bug technológia - barát vagy ellenség?," *Szabad adatok, védett adatok,* 2005.

[7] J. Gomez, T. Pinnick and A. Soltani, "KnowPrivacy. Technical Report 2009-037," University of California at Berkeley, 2009.

[8] A. Efrati, "'Like' Button Follows Web Users," The Wall Street Journal, 2011. 05. 18. [Online]. http://online.wsj.com/article/SB10001424052748704281504576329441432995616.html. [2012. 05. 20.].

[9] "What are local shared objects?," Adobe Inc., [Online]. http://www.adobe.com/security/flashplayer/articles/lso/. [2012. 05. 20.].

[10] A. Soltani, S. Canty, Q. Mayo, T. Lauren and C. J. Hoofnagle, "Flash Cookies and Privacy," SSRN eLibrary, AAAI.org, 2009.

[11] S. Kamkar, "Evercookie," samy.pl, 2010. 09. 20. [Online]. http://evercookie.samy.pl/evercookie. [2012. 05. 20.].

[12] Danis M., „Webes hírcsatornákban alkalmazható nyomkövetési technikák vizsgálata," Budapest, 2011.

[13] A. Narayanan, "Cookies, Supercookies and Ubercookies: Stealing the Identity of Web Visitors," 33 Bits of Entropy, 2010. 02. 18. [Online]. http://33bits.org/2010/02/18/cookies-supercookies-and-ubercookies-stealing-the-identity-of-web-visitors/. [2012. 05. 20.].

[14] A. Narayanan, "Ubercookies Part 2: History Stealing meets the Social Web," 33 Bits of Entropy, 2010. 02. 19. [Online]. http://33bits.org/2010/02/19/ubercookies-history-stealing-social-web/. [2012. 05. 20].

[15] S. Stamm, "Plugging the CSS History Leak," Mozilla.org, 2010. 03. 31. [Online]. http://blog.mozilla.org/security/2010/03/31/plugging-the-css-history-leak/. [2012. 05. 20.].

[16] P. Eckersley, "How Unique Is Your Web Browser?," Springer, Heidelberg, 2010.

[17] K. Boda, Á. M. Földes, G. G. Gulyás and S. Imre, "User Tracking on the Web via Cross-Browser Fingerprinting," in *16th Nordic Conferene in Secure IT System (Nordsec 2011)*, Talinn, Estonia, 2011.

[18] L. Sweeney, "Simple Demographics Often Identify People Uniquely," Pittsburgh, 2000.

[19] G. Aggarwal, E. Burzstein, C. Jackson and D. Boneh, "An Analysis of Private Browsing Modes in Modern Browsers," *Proceedings of the 19th USENIX Security Symposium*, 2010.

[20] D. Chaum, "Untraceable electronic mail, return addressres, and digital pseudonyms," *Communications of the ACM*, pp. 84-90, 1981.

[21] R. Dingledine, N. Mathewson and P. Syversorn, "Tor: The Second-Generation Onion Router," *13th USENIX Security Symposium*, 2004.

[22] Gulyás G. Gy. és Schulcz, R. „Újgenerációs anonim böngészők," *Híradástechnika,* pp. 24-27, 2007/8.

[23] Gulyás G. Gy., *Anonimitás a weben.* [Előadás]. 2012. 03. 29.

Mellékletek

Táblázatok

Az ujjlenyomat módszerek technikai háttere

A JavaScript API-n keresztül elérhető releváns adatok teljes táblázata. Az *adat elérése* mezőben a zárójelben lévő megjegyzés arra utal, hogy csak a zárójelben felsorolt böngészőprogramokból érhető el az adat.

Adat elérése	**Leírás**	**Példa**
navigator.userAgent	User Agent String (böngészőleíró)	`Mozilla/5.0 (Windows NT 6.1; WOW64; rv:11.0) Gecko/20100101 Firefox/11.0`
navigator.appVersion	Mozilla verzió (UAS alapján minden böngészőnél, nem releváns)	`5.0 (Windows)`
navigator.appCodeName	A böngésző kódneve (UAS alapján minden böngészőnél, nem releváns)	`Mozilla`
navigator.appName	A böngésző alkalmazás neve (nem releváns)	`Netscape`
navigator.platform	A böngésző fordítási platformja	`Win32`
navigator.oscpu (FF)	A futtató operációs rendszer típusát azonosítja	`Windows NT 6.1; WOW64`
navigator.language	Nyelvi beállítás kódja	`hu-HU`

Adat elérése	Leírás	Példa

navigator.plugins
(FF, CH, OP, SA)
	A telepített kiegészítők listája. Tartalmazza a kapcsolódó média típusokat (számmal elnevezett objektumokként) és a lista hosszát, ezen túl a plugin leírását, a futtatott fájl nevét és részletes verzióját, valamint a plugin nevét.	A lista egy eleme:
		0: {
		description: Adobe Flash movie
		enabledPlugin: #ref
		suffixes: swf
		type:
		application/x-shockwave-flash
		}
		1: {
		description: FutureSplash movie
		enabledPlugin: #ref
		suffixes: spl
		type: application/futuresplash
		}
		length: 2
		description:
		Shockwave Flash 11.2 r202
		filename: NPSWF32_11_2_202_228.dll
		version: 11.2.202.228
		name: Shockwave Flash
		item: [native code]
		namedItem: [native code]

navigator.mimeTypes
(FF, CH, OP, SA)
	A böngésző által értelmezett média típusok listája. Tartalmaz egy leírást, a kapcsolódó kiegészítő objektumot (enabledPlugin), a hozzá tartozó fájlnév kiterjesztéseket (suffixes), valamint a média típusát.	A lista egy eleme:
		description: Adobe Flash movie
		enabledPlugin: {
		0: #ref
		1: {
		description: FutureSplash movie
		enabledPlugin: #ref
		suffixes: spl
		type: application/futuresplash
		}
		length: 2
		description:
		Shockwave Flash 11.2 r202
		filename:
		NPSWF32_11_2_202_228.dll
		version: 11.2.202.228
		name: Shockwave Flash
		item: [native code]
		namedItem: [native code]
		}
		suffixes: swf
		type: application/x-shockwave-flash

navigator.buildID (FF)	Másodperc pontosságú „építési” azonosító.	20120312181643
navigator.doNotTrack (FF, IE, OP, SA)	Nyomkövetés elleni kérelem.	unspecified
navigator.geolocation (FF, CH, OP, SA)	GeoLocation alapú helymeghatározó szolgáltatás. Általában a felhasználó engedélye szükséges hozzá.	getCurrentPosition()
navigator.product (FF, CH, SA)	A layout engine elnevezése (UAS alapján).	Gecko
navigator.productSub (FF, CH, SA)	Nap pontosságú böngésző építési azonosító (UAS-ban is szerepel).	20100101

Adat elérése	Leírás	Példa
navigator.cookieEnabled	Cookie engedélyezés állapota.	`true`

screen, window.screen	A monitor beállításainak tényleges, illetve a böngésző számára elérhető méreteit, és a színmélység értékét tartalmazza. Kiszámítható belőle a tálca mérete.	`height: 900`
		`width: 1600`
		`pixelDepth: 24`
		`colorDepth: 24`
		`availWidth: 1600`
		`availHeight: 860`

Date.getTimezoneOffset	Időzóna eltérés a rendszer beállítása alapján.	`-120`

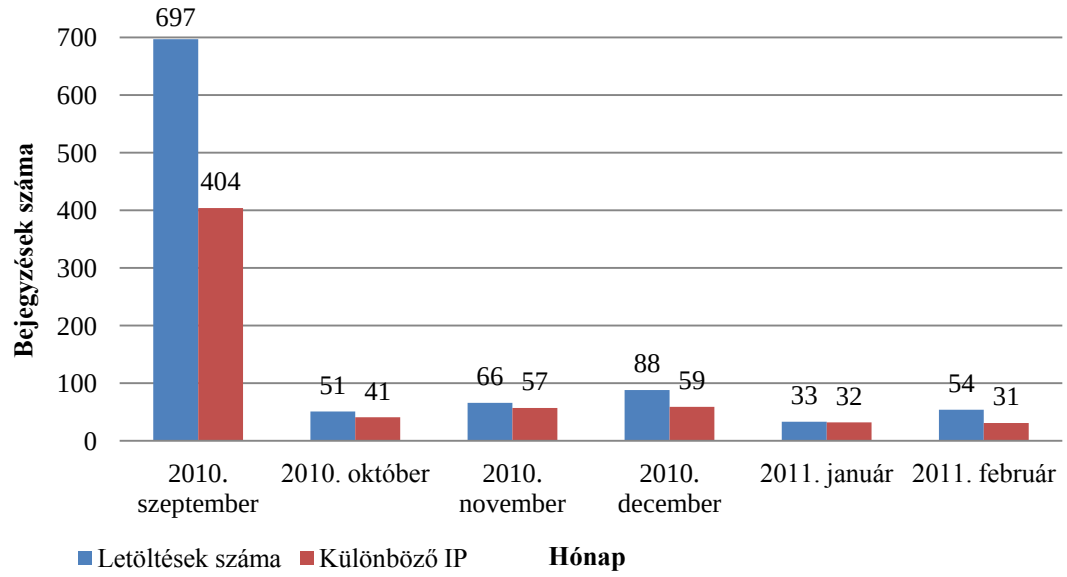
performance.timing (FF, CH, IE)	A teljesítmény mérésére szánt fejlesztői eszköz. Célja a Date objektummal való számítás kiváltása annak nagy overheadje[90] miatt, valamint az optimalizálást elősegítő részletesebb időfelbontás. Mindazonáltal remek lehetőséget ad a kliens gép teljesítményének tesztelésére (letöltési idő, renderelési idő is számítható belőle).	Részlet a listából:
		`navigationStart: 1334845694024`
		`unloadEventStart: 1334845694025`
		`unloadEventEnd: 1334845694030`
		`redirectStart: 0`
		`redirectEnd: 0`
		`fetchStart: 1334845694025`
		`domainLookupStart: 1334845694025`
		`domainLookupEnd: 1334845694025`
		`connectStart: 1334845694025`
		`connectEnd: 1334845694025`
		`requestStart: 1334845694025`
		`responseStart: 1334845694025`
		`responseEnd: 1334845694025`
		`domLoading: 1334845694025`
		`domInteractive: 1334845694348`
		`...`

[90] Az aktuális idő lekéréséhez mindig új Date objektumot kell létrehozni, a létrehozás ideje viszont hozzáadódik a mért időhöz, ami milliszekundum nagyságrendben sokat számít.

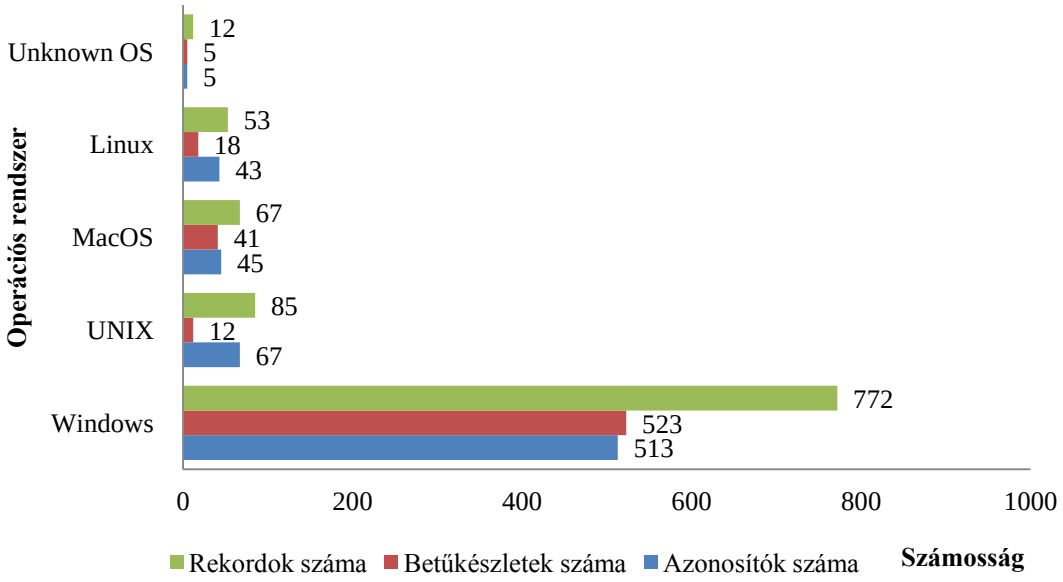
Ábrák

Statisztika (Rendszerujjlenyomat teszt)

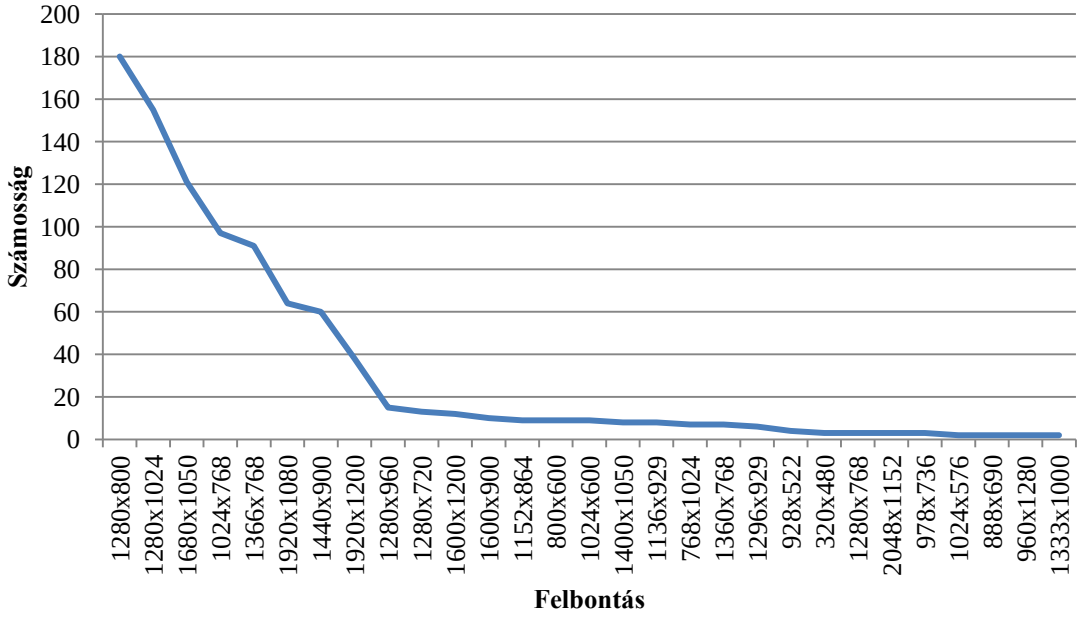
Látogatottsági statisztika havi bontásban



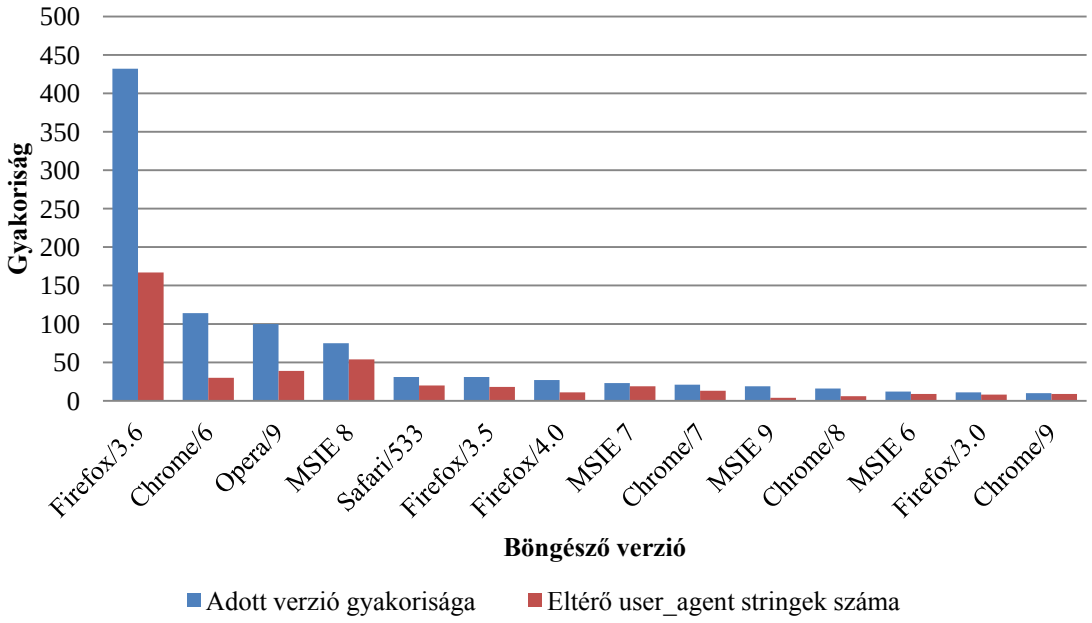
Betűtípusok és azonosítók számossága operációs rendszer szerint



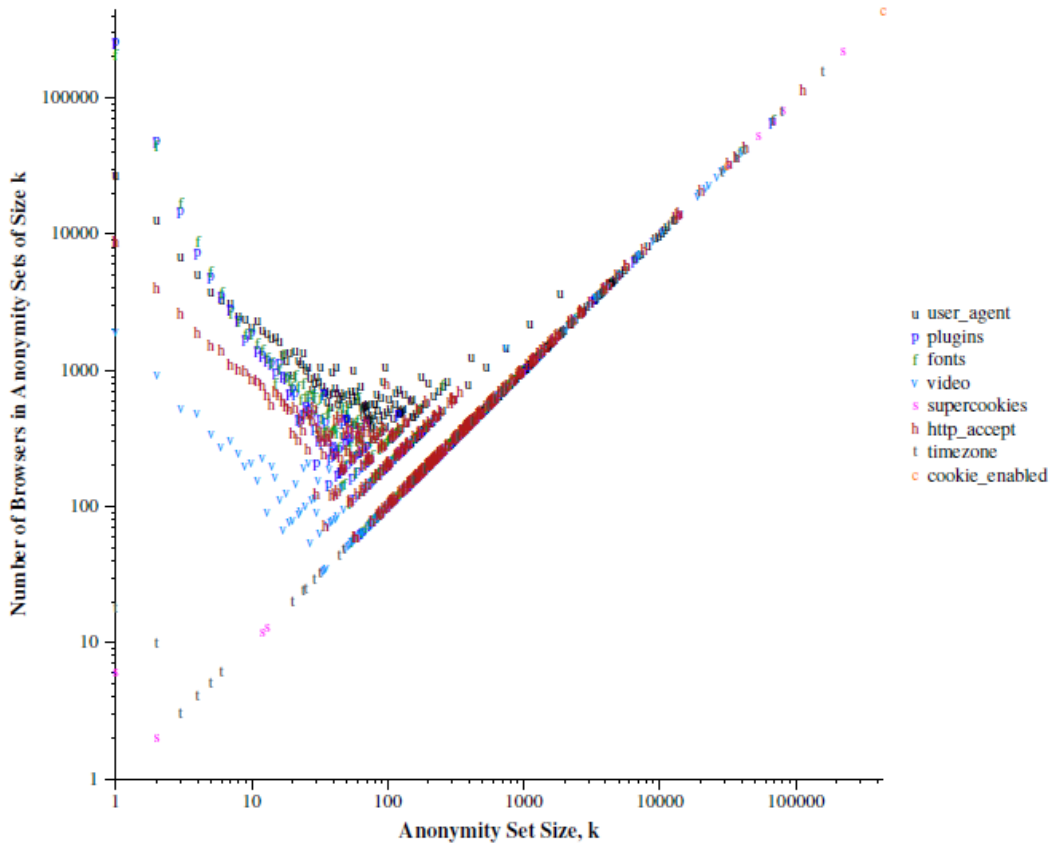
Képernyőfelbontások (top 30)

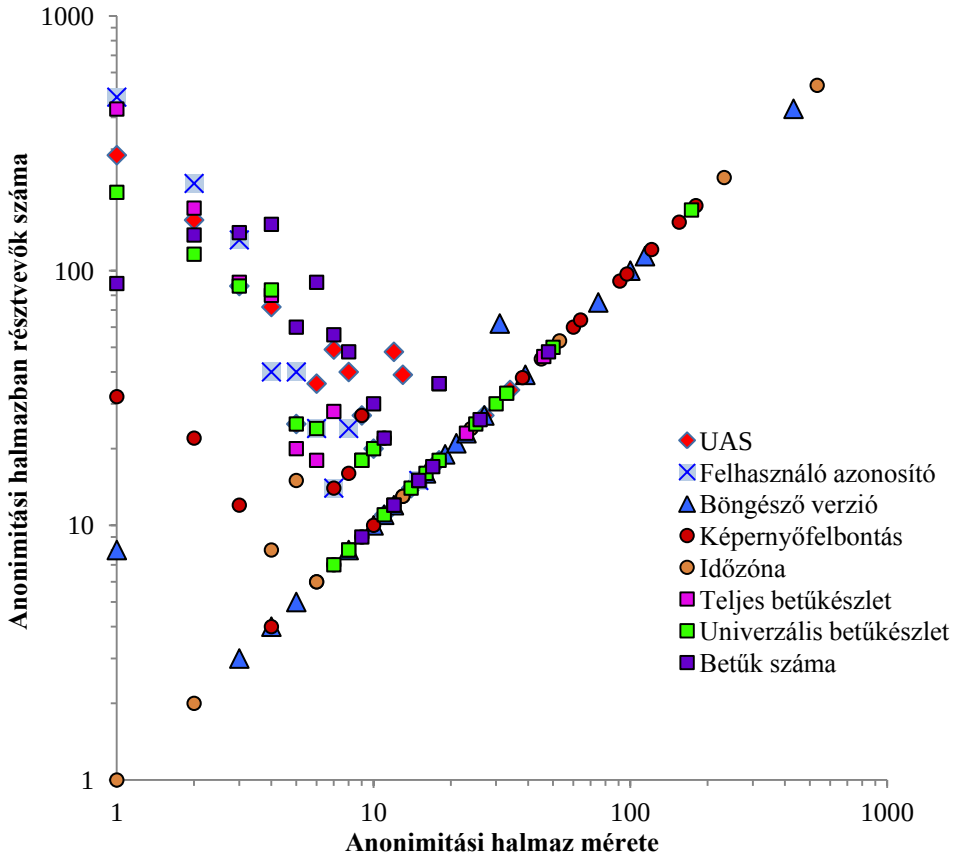


Böngészőverziók (top 15)



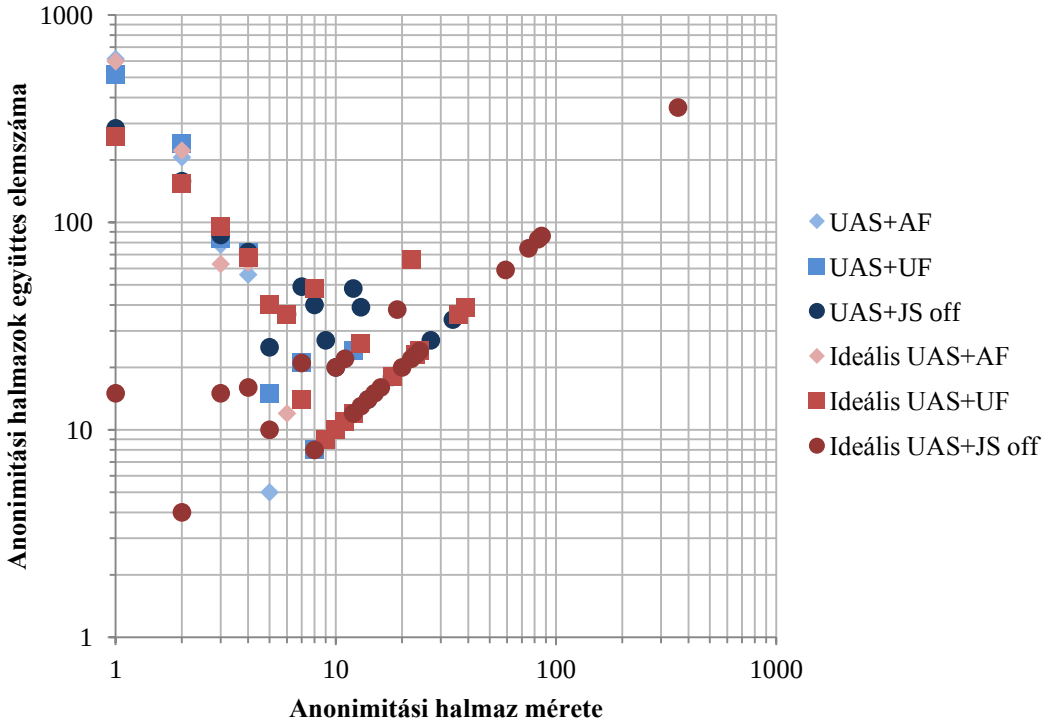
Összevetés a Panopticlick projekttel





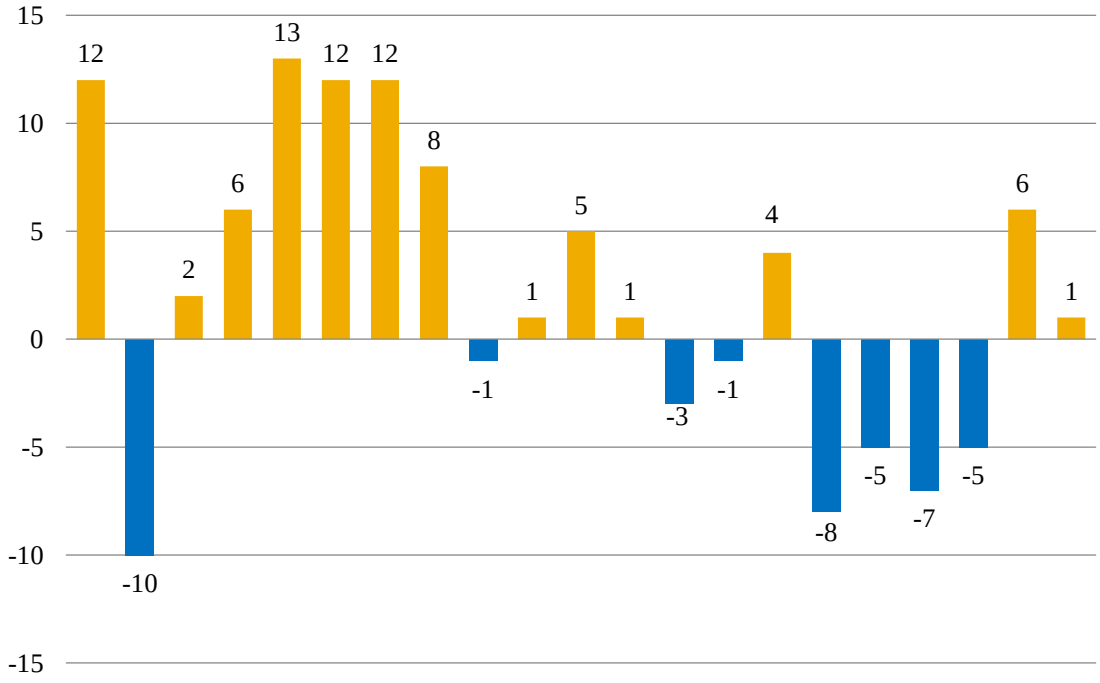
Anonimitási halmazok

Az UAS, ideális UAS, AF és UF kombinációk anonimitási halmazok mérete szerint. Látható, hogy 40 méretű anonimitási halmaznál nagyobb csak ideális UAS használatával és JS kikapcsolt állapotával érhető el. Az is leolvasható, hogy a felhasználók túlnyomó része minden attribútumot tekintve 10-nél kisebb méretű halmazba tartozik (UAS+AF).



Betűtípuslisták

Felhasználónkénti eltérő számú betűtípus detektálások számosság különbsége Firefox (narancssárga) és Internet Explorer (kék) esetén



A detektált betűtípusok számának meghatározásához a *fingerprints* táblát bővítettem egy *fonts_count* mezővel, amelybe a teljes bináris fontlista (*fonts_bin*) 1-eseinek száma került. A mező feltöltéséhez a *relations* táblát használtam, ami azokat a betűtípusokat tartalmazza minden bejegyzéshez, ahol a bináris listában 1-es szerepel. A *fonts_count* tehát a *relations* tábla *fingerprint_id* (idegenkulcs) attribútuma szerinti csoportosításával kapott csoport számosságok értéke. (Lekérdezés: ld. *fonts_count* mező kitöltése)

Lekérdezések (válogatás)

Statisztika

Látogatottság havi bontásban

```
SELECT
  DATE_FORMAT(`created`, '%Y.%m') AS date,
  COUNT( * ) AS count,
  COUNT( DISTINCT `ip` ) AS IPcount
FROM `fingerprints`
GROUP BY date
ORDER BY date ASC
```

Betűtípusok és azonosítók számossága operációs rendszer szerint (kiegészítve a bejegyzések számával)

```
SELECT subq.*, (fontsBinCount/userIdCount) AS 'Ratio'
FROM
(
SELECT
  `os`,
  COUNT( DISTINCT `user_id` ) AS userIdCount,
  COUNT( DISTINCT `fonts_bin` ) AS fontsBinCount,
  COUNT( * ) AS numrows
FROM `fingerprints`
GROUP BY `os`
ORDER BY userIdCount DESC
) AS subq
```

Képernyőfelbontások

```
SELECT `screen`, COUNT( * ) AS count
FROM `fingerprints`
GROUP BY `screen`
ORDER BY count DESC
```

Összevetés a Panopticlick projekttel

Az egyes attribútum értékek entrópiája

```
SELECT SUM((-1)*hpi*LOG2(hpi)) AS sum
FROM
(
  SELECT (cnt / (SELECT COUNT(*) FROM `fingerprints`)) AS hpi
  FROM
  (
    SELECT
    COUNT( < változó > ) AS cnt
    FROM `fingerprints`
    GROUP BY < változó >
  ) AS subq
) AS subq2
```

Betűtípuslisták

Betűtípusok számossága különböző platformokon

```
SELECT
  `title`,
  COUNT( DISTINCT `ip` ) AS distinctIpCnt,
  COUNT( DISTINCT `fonts_bin` ) AS distinctFontsBinCnt
FROM `relation` AS rel
INNER JOIN `fingerprint_fonts` AS fonts
  ON rel.`fingerprint_font_id` = fonts.`id`
INNER JOIN `fingerprints` AS fprints
  ON fprints.`id` = rel.`fingerprint_id`
WHERE `os` = < operációs rendszer >
GROUP BY `fingerprint_font_id`
ORDER BY distinctIpCnt DESC
```

A fonts_count mező kitöltése

```
UPDATE `fingerprints`,
(
  SELECT `fingerprint_id`, COUNT( `fingerprint_id` ) AS cnt
  FROM `relation`
  GROUP BY `fingerprint_id`
) AS subq
SET `fonts_count` = subq.cnt
WHERE `id` = subq.`fingerprint_id`
```

Eltérő betűtípus detektálások Firefox és Internet Explorer esetén

```
SELECT
  subq_ff.`ip`,
  subq_ie.`fonts_count` AS ie_count,
  subq_ff.`fonts_count` AS ff_count,
  ABS(subq_ie.`created` - subq_ff.`created`) AS timediff,
  subq_ie.`user_agent` AS ie_ua,
  subq_ff.`user_agent` AS ff_ua
FROM
(
  SELECT `ip`, `fonts_count`, `user_agent`, `user_id`, `created`
  FROM `fingerprints`
  WHERE `user_agent` LIKE '% MSIE%'
) AS subq_ie,
(
  SELECT `ip`, `fonts_count`, `user_agent`, `user_id`, `created`
  FROM `fingerprints`
  WHERE `user_agent` LIKE '% Firefox%'
) AS subq_ff
WHERE
  subq_ie.`ip` = subq_ff.`ip` AND
  subq_ie.`user_id` = subq_ff.`user_id`
HAVING timediff < 3600 AND ie_count <> ff_count
ORDER BY ie_count DESC
```

Speciális esetek vizsgálata

Egyező összetevők, eltérő azonosítók listázva

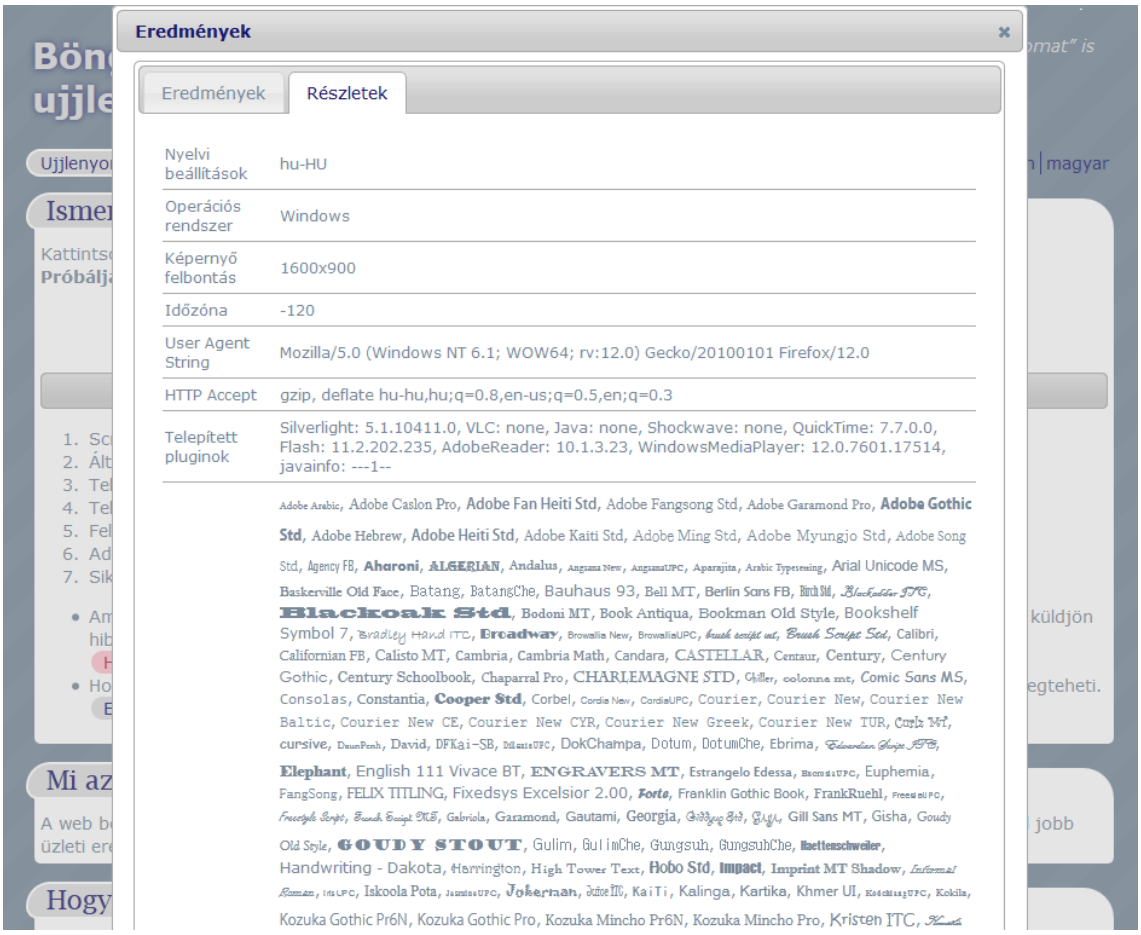
```
SELECT
   `user_id` , `created` ,
   `subq`.`ip`    ,    `subq`.`screen`    ,    `subq`.`timezone`    ,
   `subq`.`fonts`
FROM
   `fingerprints`,
   (
   SELECT `ip` , `fonts` , `screen`, `timezone`,
   COUNT( DISTINCT `user_id` ) AS count
   FROM `fingerprints`
   GROUP BY `ip` , `fonts` , `screen` , `timezone`
   HAVING count > 1
   ORDER BY count DESC
   ) AS `subq`
WHERE
   `fingerprints`.`ip` = `subq`.`ip` AND
   `fingerprints`.`fonts` = `subq`.`fonts` AND
   `fingerprints`.`screen` = `subq`.`screen` AND
   `fingerprints`.`timezone` = `subq`.`timezone`
GROUP BY `user_id`
```

Képernyőképek

A Böngészőfüggetlen ujjlenyomat teszt 2.0 weboldala

Párbeszéd ablak a becenév megadásához

A detektálás részleteit megjelenítő fül az Eredmények ablakban



Üzenetküldő felület

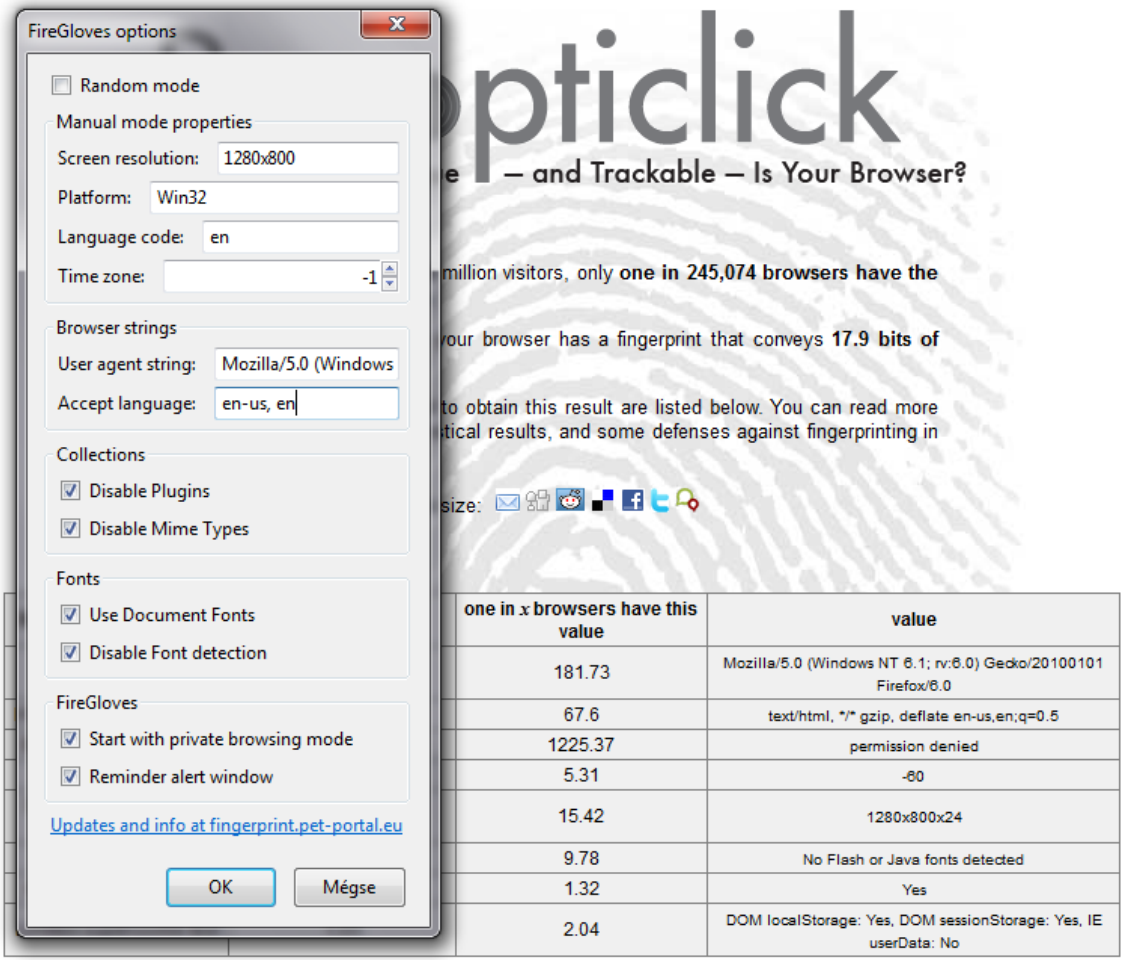
Adminisztrációs beléptető felület

Menüelemek listája az adminisztrációs felületen

Cikk jellegű tartalom szerkesztése az adminisztrációs felületen

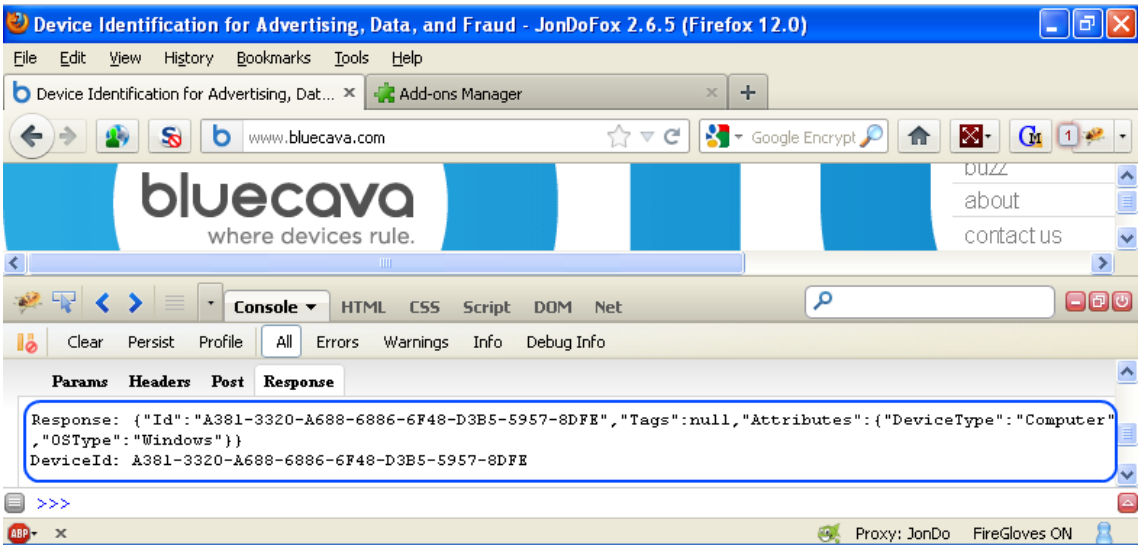
A FireGloves működés közben

Panopticlick: a kiegészítő meghamisította az adatokat, és letiltotta a plugin lista elérését, ezért a teszt sem a betűtípusokat sem a pluginlistát nem érte el.



FireGloves és JonDoFox együttes használatával, azonos beállításokkal generált BlueCava azonosítók Windows XP és Ubuntu rendszereken (különböző IP címről): sikerült ugyanazt az azonosítót létrehozni.

Windows XP:



Ubuntu: