Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Híradástechnikai Tanszék

Kliensoldali webes tartalomtitkosító eszköz készítése

Készítette: Paulik Tamás
Tanszéki konzulens: Gulyás Gábor György, Schulcz Róbert

2009.

Nyilatkozat

Alulírott Paulik Tamás, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem el és abban csak az irodalomjegyzékben felsorolt forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva a forrásokból átvettem, egyértelműen, a források megadásával megjelöltem.

Budapest, 2009. december 8.

.....

Paulik Tamás

Köszönetnyilvánítás

Ez úton szeretném megköszönni Gulyás Gábornak a diplomaterv készítésében nyújtott segítséget, hogy egy, a valós életben is fontos, figyelmet igénylő témával foglalkozhattam és egy olyan terméket készíthettem el, amellyel ez a probléma érdemben kezelhető.

Kivonat

Az úgynevezett WEB 2.0-ás folyamat eredményeként megjelenő új szolgáltatások vonzereje komoly üzleti értékkel ruházta fel az internetet. Online üzletek, reklámok és egyéb szolgáltatások találhatóak bármerre, amerre a felhasználó jár. A blogok, a mikroblogok, a fórumok, a kép- és videó megosztó oldalak, a szociális hálózatok csak néhányak azok közül a szolgáltatások közül melyek uralják az internetet. Közös vonzerejük, hogy azt nyújtják, amire az embernek a mai információs társadalomban szüksége van. A felhasználók hatalmas mennyiségű információhoz juthatnak hozzá, gyakorlatilag bármiről és az előbb felsorolt interaktív szolágáltatásokon keresztül ők maguk is teremthetnek tartalmat, megmutathatják kreativitásukat, képességüket, örömüket-bánatukat.

Ez a nagy mennyiségű információ hatalmas üzleti értéket képvisel, szolgáltatók és cégek alakultak kizárólag avval a céllal, hogy megfigyeljék a felhasználókat, profilt építsenek róluk, amiket később eladhatnak másoknak, hogy azok ezen információk alapján fejleszthessék saját szolgáltatásukat. Sajnos ezek az információk visszaélésekre is alkalmat adnak, a profilinformációk felhasználhatóak a felhasználói igények kielégítésén kívül a felhasználók megkárosítására is. Alkalmat adnak a célzott adathalászatra, személyre szabott jelszótörő könyvtár építésére, az információ felhasználhatósága csak a támadó kreativitásán múlik.

Jelen dolgozat célja egy olyan szoftver tervezése és implementációja, mellyel a felhasználók megvédhetik privátszférájukat az ilyen támadásokkal szemben. A szoftver lehetőséget ad a weboldalakon bevitt szöveges információk titkosítására, így segítve elő az információ biztonságát és biztosítva a hozzáférés vezérlést ott, ahol az másképp nem megoldható.

Abstract

As the result of the so-called WEB 2.0 process, the new online services brought previously unrevealed business possibilities to the Internet. Online shops, advertisements and other services can be found everywhere, wherever users crawl. The blogs, microblogs, forums, picture- and video sharing, social networking sites are just some examples from the huge list of the services, dominating the Internet. They provide the something valuable needed by the information society. Users can get access to huge amount of information, almost about everything, and with these interactive services they can provide their own content for showing their creativity, knowledge, joy, or sorrow.

This great amount of information has a huge business value, wherefore service providers and firms were founded just for monitoring and profiling users, to build, sell user profiles to other services, for improving their services according to this information. Unfortunately this gives an opportunity to abuse the information, and the profile could be used to damage users' privacy. As it's making the targeted phishing attacks easier, it also gives the chance to build personal dictionaries for password breakers, and only the creativity of the attacker can limit the opportunities.

The goal of this thesis is to design and develop a software, which helps the users defending their privacy against such attacks. The program gives the opportunity to encrypt textual information by making the information more secure, and placing the access control in the hands of the users on most of the common online services.

Tartalom

Rövidítésjegyzék

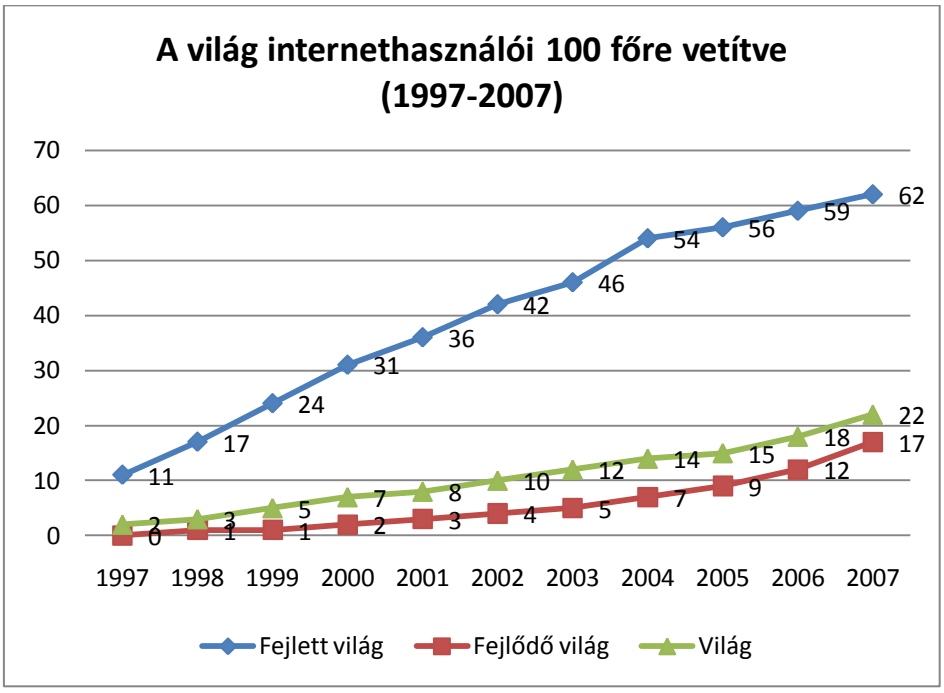
Rövidítés	Feloldás	Magyarázat
API	Application Programming Interface	A szolgáltatás azon interfésze, melyen keresztül a programozó igénybe veheti szolgáltatásait
CGI	Common Gateway Interface	A CGI egy protokoll, amely megadja, hogy a webszervernek milyen módon kell átadnia az oldal elkészítésének feladatát egy konzol alkalmazásnak
CMS	Content Management System	Tartalomkezelő rendszer. Olyan komplex weboldal, vagy portál, amely több féle tartalmat képes kezelni és ezekkel kapcsolatos szolgáltatásokat nyújtani a felhasználók felé
DRM	Digital Rights Management	Azon módszerek gyűjtőneve, melyek a digitális tartalmak hozzáférés vezérlését végzik
LSO	Local Shared Object	Olyan állomány, melyben a Flash programok tárolhatnak információkat a felhasználó számítógépén
FTP	File Transfer Protocol	Egy fájlok átvitelére használt protokoll
GNUPG	GNU Privacy Guard	Egy privátszféra védelemmel foglalkozó program csomag
HTML	Hipertext Markup Language	A weboldalak struktúrájának leírásához használatos nyelv
NAT	Network Adress Translation	Egy olyan eljárás, melynek segítségével több kliens használhat egy hálózatot olyan módon, hogy a hálózat irányából nézve csak egy IP címmel rendelkeznek
PET	Privacy Enhancing Technology	Privátszférát erősítő technológia, olyan megoldás, mely a felhasználó privátszférájának védelmére jött létra
PGP	Pretty Good Privacy	Adatok titkosítására és aláírására használatos programcsomag
WYSIWYG	What You See Is What You Get	Azon szövegszerkesztőket szokták evvel a jelzővel illetni, ahol a monitoron látható kép, megegyezik a nyomtatottal

1 Az internet elterjedésével megszülető új veszélyforrások

A ma emberének hétköznapi élete szorosan kapcsolódik az internethez, egyre nagyobb mértékben függünk tőle. Az internet térhódítása szükségessé teszi, hogy foglalkozzunk veszélyeivel, használatának kockázataival.

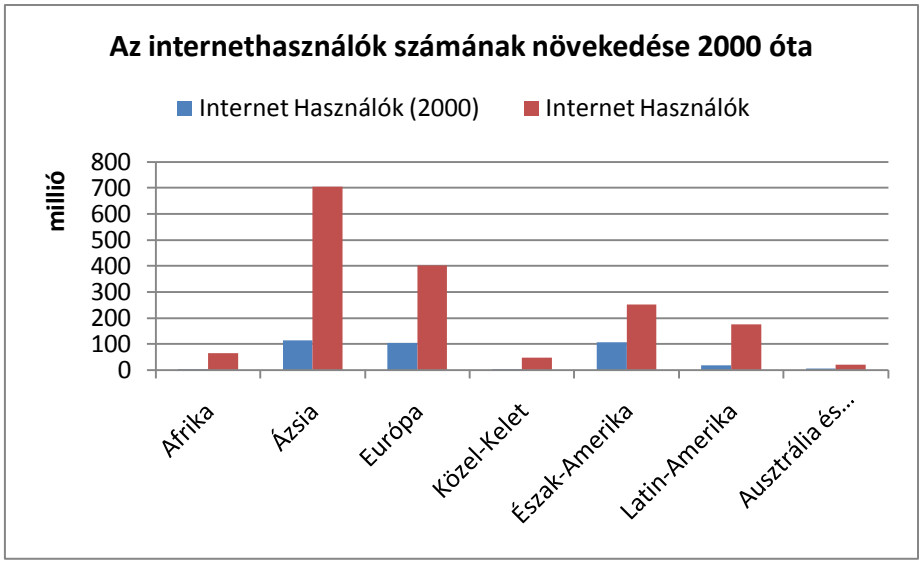
1.1 Az internet fejlődése és a Web 2.0

Az internet világának fejlődésével a felhasználók egyre újabb és újabb lehetőségekhez jutnak, a böngészés a mindennapok szerves része lett, több háztartásban az internetelérést a vízzel, gázzal és árammal egyenrangú közüzemi szolgáltatásként tartják számon. Az internetelérések egyre olcsóbbak és gyorsabbak, az internet-felhasználók száma az elmúlt néhány évben nagy mértékben megnőtt.



1-1. ábra - A 100 főre jutó internethasználók száma (1997-2007)[1]

Mindennek elsődleges oka az interneten alkalmazott technológiák színesedése, az általuk megvalósított szolgáltatások felhasználó-közelibbé válása. Az internet elterjedésének kezdetén a weben jellemzően statikus tartalmak voltak jelen, melyen nem voltak képesek alkalmazkodni a felhasználói, szolgáltatói igényekhez fejlesztői beavatkozás nélkül.



1-2. ábra - Az internethasználók számának növekedése 2000 óta[1]

A későbbiekben a CGI és a PHP és egyéb dinamikus oldal generáló technológiák elterjedése meghozta az internet forradalmát, olyan új szolgáltatások jelentek meg, melyek merőben különböztek a megszokottaktól, az internetet újszerűvé, kívánatossá tették azok számára is, akik eddig nem használták azt. Megjelentek azok az oldalak, melyek dinamikusan reagáltak a felhasználói igényekre, tartalmuk az üzemeltető beavatkozása nélkül is frissülhetett, változhatott, a felhasználók maguk is részt vehettek az oldal életében, az általuk írt fórumbejegyzések, hozzászólások beépültek az oldalba, elérhetővé váltak mások számára, a felhasználó egy új lehetőséget kapott, az információ publikálásának lehetőségét.

Az ezredfordulóra ezek a változások olyannyira megváltoztatták az internet jellegét, hogy megszületett a Web 2.0 fogalma[2], amely már magába foglalta, sőt alapelemként kezelte a nemrégiben megszületett interaktivitást és a felhasználói információkat. Elsődleges értékké maga a felhasználó és a rajta keresztül elérhető bevétel vált. Minél inkább kielégíti egy felhasználó igényeit egy weboldal, annál több bevételre számíthatnak az üzemeltetők az elhelyezett reklámokból, áruba bocsájtott termékekből, szolgáltatásokból. Ehhez azonban minél jobban meg kell ismerni a felhasználót, valamint a felhasználó által generált tartalom is alapja lehet a szolgáltatásnak.

1.2 Jellemző Web 2.0-ás szolgáltatások

A szolgáltatók érdeke tehát azt kívánja, hogy olyan szolgáltatásokat nyújtsanak, melyek információ megosztásra sarkallják a felhasználókat. A ma tipikus szolgáltatója csak egy

[1] forrás: Internet World Stats - http://www.internetworldstats.com/stats.htm

keretrendszert biztosít, amelyben a felhasználó képes minden szaktudás nélkül létrehozni saját oldalát, és használni azt. Az oldal beüzemeléséhez nem szükséges a hozzáértő segítség, a szolgáltatás már az oldal elkészítéséhez is szemléletes, könnyen használható felületet biztosít. Megszülettek többek között az alábbi merőben újszerű és ötletes, ugyanakkor sokszor hasznos, vagy csak szórakoztató szolgáltatások és szolgáltatási kategóriák.

Szolgáltatás	Példák
Közösségi oldalak	iWiW[2], Orkut, Facebook[3], MySpace, LinkedIn, Mindenki.hu, Barátikör.com, MyVIP[4]
Képmegosztó oldalak	Flickr, Indafotó, Picasa, Photobucket, SmugMug, Zooomr, Open Photo Project
Videómegosztó portálok	YouTube, Google Videos, IndaVideó
Blogok, mikroblogok	Twitter[5], Jaiku.com, Plurk, Blog.hu[6]
Online irodai alkalmazások	Google Calendar, Google Docs & Spreadsheets, Zoho
Szabadon szerkeszthető ismerettárak	Wikipédia
Fórumok	SG fórum
Aukciós oldalak	eBay, Vatera
Online kiskereskedelemmel foglalkozó vagy azt támogató oldalak	PayPal, Abaqoos
Linkmegosztó szolgáltatások	del.icio.us, Diigo, kedvenceim.com, mylink.hu, Ma.gnolia.com

[2] A szolgáltatás címe: http://www.iwiw.hu

[3] A szolgáltatás címe: http://www.facebook.com

[4] A szolgáltatás címe: http://www.myvip.hu

[5] A szolgáltatás címe: http://www.twitter.com

[6] A szolgáltatás címe: http://www.blog.hu

Szolgáltatás	Példák
Más oldalakat értékelő szolgáltatások,	Digg
Hírforrások (feedek) (RSS, Atom) és a hozzá kapcsolódó szolgáltatásokat nyújtó alkalmazások	Yahoo Pipes, AideRSS, FeedHub,, FeedRinse, FeedBlendr
Hírek újrakeverését (remixelését) támogató egyéni kezdőlapok	iGoogle, Netvibes, Hírfigyelő Google Reader
Közösségi zeneajánló/szolgáltató oldalak	Pandora.com, Last.fm
Online tárhely-szolgáltatók	Box.net, Dropbox
Virtuális világok és online játékok	Second Life
Egyesített beléptető rendszerek	OpenID, TypeKey
Online térképalkalmazások	Google Maps, Yahoo! Maps
Webes távoli asztali elérést biztosító szolgáltatások	LogMeIn
Online csevegő szolgáltatások, melyek kompatibilisek a megszokott csevegő programokkal.	EBuddy, WebMessenger

1-1. táblázat - A jellemző Web 2.0-ás szolgáltatások[7]

Ezen szolgáltatások részben a WEB 1.0-ás szolgáltatásokból nőtték ki magukat, részben pedig a megnövekedett fejlesztői lehetőségek kibontakozásakor keletkeztek. Ezt az átalakulást szemlélteti az alábbi, közel sem teljes felsorolás, amely egyes szolgáltatás-típusok metamorfózisát mutatja be.

[7] forrás: http://hu.wikipedia.org/wiki/Web_2.0

Web 1.0		Web 2.0
DoubleClick	→	Google AdSense
Ofoto	→	Flickr
Akamai	→	BitTorrent
Mp3.com	→	Napster
Britannica Online	→	Wikipedia
Személyes weboldalak	→	Blogolás
Evite	→	Upcoming.org és EVDB
Domain spekuláció	→	Kereső-optimalizálás
Oldal nézettség	→	Költség/kattintás
Screen scraping[8]	→	Web szolgáltatás
Publikálás	→	Részvétel
Tartalomkezelő rendszerek	→	Wikik
Taxonómia	→	Folkszonómia
Stickiness	→	Syndication

1-2. táblázat - A Web 1.0 átalakulása[2]

Az internet megváltozásával a felhasználókra leselkedő veszélyek is megváltoztak, újabbak jelentek meg, illetve régebbiek megújult formában jelentek meg.

1.3 A privátszféra és a Web 2.0

A felhasználó tehát központi szereplője, éltető eleme lett a webnek. A való világban mindenkit megilletnek alapvető jogok, melyek a felhasználó privátszférájára vonatkoznak, megilleti őt az adatainak védelme és az adatok hozzáférhetőségének kezelése. Az, hogy az internet a hétköznapok részévé lett, szükségessé tette, hogy ezek a jogok, igények és kötelezettségek az interneten is megjelenjenek. Kell, hogy az interneten is érvényben legyenek szabályozások, melyek a felhasználó védelmében jöttek létre és figyelmet kell fordítani ezek betartatására is. Az internet folyamatosan változó és kaotikus világában ennek a megvalósítása igen nehezen kivitelezhető, ennek köszönhető, hogy az internet rengeteg privátszféra elleni támadás színtere napjainkban.

[8] A megnevezés az eredeti szövegben is így szerepelt, az érthetőség végett nem találtam szükségesnek a lefordítását

2 A Web veszélyei privátszféra szempontból

Az internet átalakulása új veszélyeket teremtett, melyekkel a felhasználóknak meg kell küzdeniük. Mivel az internet piaccá lépett elő, ezért szükségszerűen megjelentek olyan szereplők, melyek ezeket az új lehetőségeket kihasználva próbálnak érvényesülni. Ezen támadókkal szemben elsősorban a felhasználó áll, akinek érdeke elsősorban az, hogy megőrizhesse kontrollját adatai felett, meghatározhassa, hogy mikor és mennyit oszt meg ezekből a szolgáltatásokkal azért, hogy cserébe esetlegesen maga is előnyökhöz jusson.

A továbbiakban áttekintjük ezeket a szereplőket, céljaikat, motivációikat, valamint eszközeiket és röviden felmérjük, hogy a felhasználó milyen lehetőségekkel bír, ha meg akarja védeni adatait.

2.1 A privátszféránk támadásában érdekelt felek

A felhasználóval szemben igen sok, mind-mind más és más profiligényekkel rendelkező támadó áll, akik saját céljaik elérése érdekében, építik, adják, veszik a felhasználói profilokat. A továbbiakban röviden ismertetem ezen szereplőket. [3]

2.1.1 Hirdetők

A hirdetők elsődleges célja, hogy az általuk elhelyezett hirdetésekre minél többen kattintsanak rá, hiszen ezen forgalomból származik bevételük. Ennek érdekében profilokat építenek és vesznek, megpróbálva azokat összekötni a hirdetéseket megjelenítő felhasználóval. Ha sikerül a felhasználó profilját azonosítani, akkor személyre szabott hirdetéseket jelenítenek meg számára, így növelve a hirdetésre kattintás esélyét. Amíg ezek a hirdetések megmaradnak az oldal struktúrájában diszkréten megbújó reklámok formájában, addig ez a szolgáltatás akár hasznos is lehet, azonban a hirdetők egyre gyakrabban a felhasználót erőszakos módon, felugró ablakokkal, egész képernyőt betöltő flash alapú reklámokkal árasztják el, melyek a felhasználót zavarják.

2.1.2 Webes boltok

A Webes boltok célja, hogy minél nagyobb felhasználói elégedettséget érjenek el és a felhasználókat magukhoz szoktassák. Erre a célra dinamikus árazást használhatnak, melynek segítségével a termékek árait a felhasználó vélt fizetőképességéhez próbálják igazítani, így törekedve a maximális profitra. [4] Ehhez felhasználják az előzetesen épített, vagy vásárolt profilokat, melyeket összekapcsolnak a vásárlóval és a profil alapján szabják meg áraikat.

2.1.3 Adatgyűjtők

Az adatgyűjtők önmagukban nem használják fel a profilokat, csak építik és értékesítik azokat. Elsődleges céljuk a lehető legnagyobb mennyiségű adat megszerzése, ehhez a lehető legtöbb módszert bevetik, ami rendelkezésükre áll. Alkalmazhatnak Webpoloskákat, követésre alkalmassá tett hirdetéseket, járulékos böngésző adatokat (pl. referrer), Cross-site scriptinget, vásárolhatnak statisztikákat weboldalaktól, más adatgyűjtőktől. Bevethetnek spyware alkalmazásokat a felhasználói adatok elemzésére, a lényeg, hogy minél pontosabb és részletesebb profilt készítsenek.[9]

2.1.4 Cenzúrázó és felügyelő szervek

Ezen kategória szereplőinek célja, hogy megszerezzék a kontrollt a felhasználó által közzétett tartalmak felett. Elsődlegesen munkahelyek, publikus internet elérést biztosító pontok (például kávézók, könyvtárak), vagy egyes hatóságok, kormányok próbálhatják korlátozni a felhasználót, megfigyelni tevékenységét. Céljuk lehet csupán annyi, hogy megakadályozzák a felhasználót abban, hogy hálózatukon szabályaikkal ellentétes üzenetű tartalmakat publikáljon, de lehetőségük lehet az esetlegesen általuk gyanúsnak, veszélyesnek ítélt felhasználók azonosítására és tevékenységének szankcionálására.

2.1.5 Szolgáltatók

A szolgáltatók biztosítják a támadási, megfigyelési felületet a fentebb említett szereplők számára. Az ő oldalaikon keresztül tölti le a felhasználó a követésre alkalmas elemeket, például a Webpoloskákat, vagy követő scriptet. Az esetek egy részében a szolgáltató nincsen tisztában avval, hogy az oldalán külső forrásból elhelyezett objektumokkal valaki visszaél, sokszor azonban maguk a szolgáltatók is végeznek adatgyűjtést, hogy saját működésüket hatékonyabbá, eredményesebbé tegyék, vagy, hogy értékesítsék, elcseréljék a profiladatokat.

2.2 A támadók elsődleges céljai

A támadókat több cél is motiválhatja, amikor a profilok üzleti célú alkalmazása mellett döntenek. Lehetséges, hogy a profilokat saját szolgáltatásaik vonzóbbá tételéhez kívánják felhasználni, de a profil birtoklása olyan lehetőségekkel is felruházza a támadót, amelyek a felhasználó közvetlen megkárosítására alkalmasak.

[9] Az említett technológiákról részletesen a későbbiekben

2.2.1 Profilkészítés – az elsődleges cél

A profilépítők elsődleges célja, hogy egy minél teljesebb és részletesebb profillal rendelkezzenek a felhasználóról, hogy a lehető legszélesebb körben adatokat gyűjtsenek róla és ezen adatok felhasználásával kielégítsék saját információs igényeiket, valamint a tőlük profilokat vásárló más szolgáltatókét. A begyűjtendő adatok körét általában nem korlátozzák, összegyűjtenek mindent, amihez hozzáférhetnek, megismerik a felhasználó szokásait, igényeit, céljait, lehetőségeit, preferenciáit, akár szexuális orientációját is.

2.2.1.1 Célzott hirdetések

A célzott hirdetéseket elsősorban a reklámszolgáltatók alkalmazzák, profitjuk növelése érdekében. Számukra különösen fontos, hogy olyan hirdetéseket mutassanak a felhasználónak melyekre az nagy valószínűséggel rákattint, érdeklődik utána, mert bevételüket a sikeresen elhelyezett hirdetések alapján szerzik. Ehhez igen nagy segítséget nyújtanak nekik a részletes profilok, ezekből nyerik ki azon információkat, melyek segítségével a megfelelő reklámokat képessé válnak kiválasztani.

2.2.1.2 Személyre szabott szolgáltatások

A felhasználó igényeinek ismeretében a szolgáltatónak lehetősége nyílik az általa nyújtott szolgáltatásokat személyre szabni. Ha a szolgáltatás képes igazodni a felhasználóhoz, akkor a felhasználó kényelmesebbnek, jobbnak fogja érezni azt, így gyakrabban fogja látogatni, nagyobb bevételt fog termelni. Egyszerű példa lehet egy ilyen funkció megvalósítására, hogy a szolgáltatás csak olyan híreket jelenít meg a főoldalán, melyek a felhasználó fő érdeklődési körébe tartoznak. Ez azonban magában rejti a visszaélés lehetőségét is, a szolgáltatás képes lehet például, az előző példánál maradva, eltitkolni bizonyos információkat, vagy késleltetve megjeleníteni azokat.

2.2.1.3 Dinamikus árazás

A dinamikus árazást a webes áruházak alkalmazzák. Céljuk, hogy áraikat úgy szabályozzák, hogy a felhasználók még éppen elégedettek legyenek a kiszabott árakkal, így lehetőségeikhez mérten a lehető legnagyobb profitra tegyenek szert. Tekintettel arra, hogy az internet megjelenése az árak és eladók összehasonlítását egyszerűvé tette, az üzletek számára szükségessé vált, hogy áraikat gyorsan és dinamikusan szabhassák a felhasználók igényeihez, megelőzve vetélytársaikat.

Az, hogy a felhasználók számára ilyen módon egyenlőtlen feltételeket biztosítanak, sokszor vezet felháborodáshoz és megkérdőjelezi a dinamikus árazás szükségességét a vevők részéről[4], hiszen az ő érdekük az, hogy egyenlő feltételek mellett a számukra legkedvezőbb

ajánlattal élhessenek, nem pedig az, hogy a pénztárcájukhoz mért ajánlatok közül kelljen kiválasztaniuk azt, amelyik a legkisebb mértékben károsítja őket.

2.2.2 Identitás lopás

Míg a fenti alkalmazások arra koncentráltak, hogy a felhasználó ismeretét a szolgáltatás szimpatikusabbá tételére használják fel, így növelve a profitot és csábítva magukhoz újabb vendégeket, addig jelen esetben a haszonszerzés a felhasználó megkárosításán keresztül valósul meg. A támadó megfigyeli a felhasználót, profilt épít róla, megismeri és az így szerzett információk alapján megpróbál hozzáférni az általa használt szolgáltatásokhoz, visszaélni ottani lehetőségeivel, tranzakciókat indít a nevében, vagy üzeneteket küld, esetleg bűntényeket követ el úgy, hogy közben az áldozat azonosítója mögé rejtőzik el.

2.2.3 Tömeges és célzott adathalászat

A felhasználó megkárosításának másik, igen elterjedt formája az adathalászat. Az ilyen támadások során a támadók olyan információkkal árasztják el a felhasználókat, melyek hatására azok megadják nekik hozzáférési adataikat, vagy meglátogatják az általuk ártó kóddal megfertőzött weboldalt. Ilyen lehet például, hogy bankjuk felszólítja őket, hogy sürgősen jelentkezzenek be a megadott címen, különben a számlájukat zárolják. [5] Ha támadók célja, hogy minél több felhasználót átejtsenek és kicsalják tőlük belépési adataikat, tehát elsősorban sok belépési információ megszerzése a céljuk és érdektelen számukra, hogy ezek kiktől származnak, akkor tömeges adathalászatról beszélünk (phishing). Ettől eltér a célzott adathalászat (whaling), amelynek célja egy adott személy belépési adatainak megszerzése.

Utóbbi esetben a támadók sikere azon múlik, hogy a küldött megtévesztő levélre rákattint e a célszemély. A célzottan kiküldött levél sikeressége növelhető, ha a levél tartalma felkelti a felhasználó érdeklődését és ehhez használható fel a meglévő profil.

2.2.4 Egyéb visszaélések

Természetesen az eddigieken felül a profiloknak más, akár egészen egyedi felhasználásai is lehetségesek, azonban a fentiek a legjellemzőbbek, melyekkel a felhasználó nap, mint nap szembesülhet. Ezeken felül jellemzően a felügyeleti-, cenzúrázó szervek, szolgáltatók is használhatnak például profilokat az azonosításhoz.

2.3 A nyomon követésen alapuló profilozás

A nyomon követésen alapuló profilozás elsődleges célja, hogy rögzítse a felhasználó mozgásait a weboldalon, illetve a weboldalak között, összegyűjtse és elemezze, hogy a felhasználó mikor és merre jár, mire kattint, és mi az, amit el sem olvas, így kapva képet arról, hogy mi érdekli őt, mi az, ami számára csábító, vagy képes felkelteni érdeklődését. Az évek során több olyan technika is kialakult melynek elsődleges célja, hogy ezt a fajta adatgyűjtést lehetővé tegye.

2.3.1 Eszközök

A felhasználó követésére szolgáló technikák néhány alapvető eszközre támaszkodnak, ezeken keresztül igyekeznek az összegyűjtött adatokat összekötni a felhasználóval, illetve összekötni a felhasználó különböző helyeken keletkezett profiljait.

2.3.1.1 IP cím

Az első és legtriviálisabb adat maga az IP cím. Ha a különböző oldalak rögzítik látogatóik IP címet, majd ezeket az adatbázisokat egyesítik, akkor megtudhatják, hogy melyik IP cím, mikor és melyik oldalt töltötte le. Így máris létrejönnek felhasználói profilok, melyek lehet ugyan, hogy konkrét személyhez nem köthetőek, ugyanakkor mindenképpen egy létező felhasználóhoz tartoznak. A dinamikus IP címek és a NAT[10] megjelenése azonban komoly korlátokat állított a profilok IP címek alapján történő összekötése elé, hiszen így egy felhasználóhoz több IP cím is tartozhat, de akár többhöz is tartozhat egy. Ennek a problémának a megoldására vezették be a követő sütiket.

2.3.1.2 A követő süti (tracking cookie)

Az IP cím helyett szükség volt egy másik, egyedi azonosítóra, mely lehetővé teszi az egyes böngészésekkor keletkezett profilok összekapcsolását. A követő sütik használatakor a szerver elhelyez egy sütit a felhasználó számítógépén és ebben egy egyedi azonosítót. A későbbiekben az általa tárolt profilt evvel az azonosítóval párosítja össze. Amikor a felhasználó újra meglátogatja az oldalt, akkor a süti automatikusan elküldésre kerül a szolgáltatónak, aki ilyen módon értesül az azonosítóról és folytathatja az előzőekben létrehozott profil építését. A technológia elterjedését követően természetesen megjelentek olyan megoldások, melyek ennek kivédésére szolgáltak, ez tette szükségessé, hogy lehetőség legyen a sütikkel végzett azonosítás kiváltására.

[10] Network Adress Translation: Egy olyan eljárás, melynek segítségével több kliens használhat egy hálózatot olyan módon, hogy a hálózat irányából nézve csak egy IP címmel rendelkeznek

2.3.1.3 Az LSO, azaz a flash süti

Az LSO[11]-k olyan állományok melyeket az oldalakba ágyazott flash objektumok használhatnak lokális adattárolásra. A sütik szűrésének megjelenése után a profilt építők elkezdték ezeket használni a követő sütikkel azonos funkciók megvalósítására, így jött létre a flash PIE, azaz a Persistent Identifier Element, amely tárolja az azonosítót és segíti a támadót a profilok összekapcsolásában.

2.3.2 A meglévő eszközökön alapuló követési módszerek

Az ismertetett eszközkészletre alapozva több olyan módszer is elterjedt, mely arra irányul, hogy kinyerje a felhasználótól a profilok összekapcsolásához szükséges adatokat, még akkor is, ha esetleg egy másik szolgáltató oldalán jár. Ezeknek a megoldásoknak közös jellemzője, hogy a weboldalban olyan elemeket helyeznek el, melyek külső szervereken tárolódnak, így lekérésükkor kapcsolat létesül a külső szerverrel is, így tájékoztatva azt is az oldal lekérésének körülményeiről.

2.3.2.1 Webpoloskák (Webbugok)

Az oldalakba ágyazott külső tartalmaknak különleges altípusa a Webpoloska, mely általában egy 1 pixeles átlátszó kép, és a célja az, hogy úgy biztosítsa a külső forrásból való letöltést, hogy a letöltéshez tartozó tartalmi elem rejtve marad a felhasználó elől.[6]

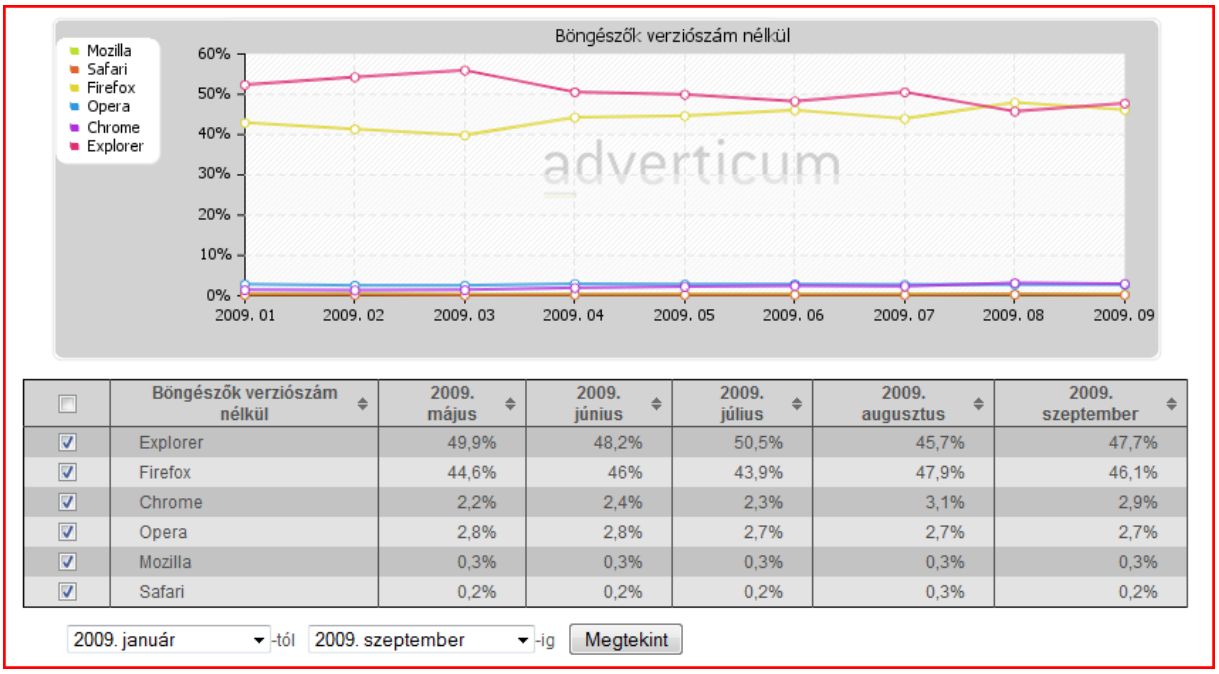
2.3.2.2 A hirdetések

Érdemes megfigyelni, hogy igen sok magyar és külföldi oldal hirdetései az Adverticum[12] hirdetés-kiszolgálótól érkeznek, így a reklámszerver értesül oldalletöltéseinkről, hiszen a reklámok az Adverticum szerverein találhatóak. Ennek ékes bizonyítéka, hogy nemrégiben elindítottak egy szolgáltatást[13], melyben a hirdetéseiken keresztül mért statisztikáik érhetőek el. Többek között a böngészők elterjedtsége, verziószámokkal, vagy aggregáltan, flash, silverlight verziók, képernyőfelbontások, operációs rendszerek. A 2-1. ábrán például a verziószámok nélküli böngésző statisztika látható, a 2009-es évre vetítve.

[11] Local Shared Object: Olyan állomány, melyet a flash objektumok használhatnak lokális adattárolásra

[12] A szolgáltató oldalai az alábbi címeken érhetőek el www.adverticum.hu és www.adverticum.com

[13] A szolgáltatás elérhető a http://adstat.adverticum.hu/ címen



Böngészők verziószám nélkül	2009. május	2009. június	2009. július	2009. augusztus	2009. szeptember
Explorer	49,9%	48,2%	50,5%	45,7%	47,7%
Firefox	44,6%	46%	43,9%	47,9%	46,1%
Chrome	2,2%	2,4%	2,3%	3,1%	2,9%
Opera	2,8%	2,8%	2,7%	2,7%	2,7%
Mozilla	0,3%	0,3%	0,3%	0,3%	0,3%
Safari	0,2%	0,2%	0,2%	0,3%	0,2%

2009. január -tól 2009. szeptember -ig Megtekint

2-1. ábra - A böngészők elterjedtsége az AdStat alapján

Ezek a reklámok lehetnek az oldalba ágyazott képek, iframe-ek, vagy flash objektumok, attól függően, hogy a hirdetést tartalmazó felülethez, vagy egyéb szolgáltatói igényekhez melyik illeszkedik jobban. A reklámszolgáltató tehát olyan hirdetéseket tesz elénk, melyeket profilunk alapján számunkra kedvezőnek ítél, és ezeket a reklámokat is felhasználja profilunk építésére.

2.3.2.3 Hivatalos Webpoloskák

Egyes szolgáltatók[14] bevallottan követik felhasználóikat és adatokat gyűjtenek róluk. Ehhez a webpoloskáknál megismert technológiákat használják, így figyelik meg a felhasználókat saját szolgáltatásaikon belül, illetve a velük együttműködő szolgáltatásokon. Ezek a szolgáltatások általában már regisztrációkor elfogadtatják a felhasználóval a követés tényét, olyan módon, hogy azt a felhasználási feltételekbe, vagy az adatvédelmi nyilatkozatba foglalják, melynek elfogadása alapfeltétele a szolgáltatás igénybe vételének.[7]

2.3.3 Egyéb módszerek a követésre

Ezek a módszerek elsősorban arra koncentráltak, hogy a kiemelt információk (IP cím, sütik, LSO-k) kinyerését megkönnyítsék. Ezeken felül a támadó egyéb információkhoz is hozzájuthat, így színesítve a profilt.

[14] Ilyen szolgáltatás például a Yahoo Beacons is. (http://info.yahoo.com/privacy/us/yahoo/webbeacons/)

2.3.3.1 A böngésző által átadott adatok

Amikor felkeresünk egy oldalt, böngészőnk automatikusan információkat oszt meg a szerverrel. Ezeket az információkat a szerver eltárolhatja és profilunkhoz kapcsolhatja. A kinyerhető információk mennyiségét jól mutatja, az IPAdressLocation.org[15] és az ProjectIP.com[16] szolgáltatása melyek listázzák ezeket az információkat. (A projectip.com által kilistázott adatok megtalálhatóak a mellékletek között.

2-2. ábra - IpAdressLocation.org

Megfigyelhető, hogy a komolyabb eszközkészletet bevető ProjectIP mennyivel nagyobb mennyiségű adatot szolgáltat.

2.3.3.2 URL referrer

Amikor egy oldalon egy linkre kattintunk, akkor a cél oldal a kérés referrer mezőjén keresztül értesül arról, hogy mely oldalról érkeztünk. Ezen hivatkozásokon keresztül követhető a felhasználó útja a weben, csak visszafelé kell elindulni a referrer-láncon. Ma már léteznek olyan szolgáltatások[17], melyek a referrer mező törlésével eltávolítják ezt az információt.

2.3.3.3 Több kiszolgáló logjainak összehasonlítása

Szintén alkalmat adhat a felhasználó követésére, ha több szolgáltató összefog, és böngészési naplóállományaikat összevetik. Megvizsgálhatják például, hogy a felhasználó melyik oldalon meddig volt aktív.

[15] A szolgáltatás elérhető a http://www.ipaddresslocation.org/ címen

[16] A szolgáltatás elérhető a http://www.projectip.com címen

[17] Ilyen szolgáltatást nyújt például az Anonym.to is. (http://anonym.to/)

2.3.3.4 Spyware

A számítógépünkön tárolt adatok, jelszavak, előzmények, sütik és LSO-k felhasználásával a szolgáltatónak lehetősége támadna egy kiemelkedően pontos profil építésére. Ehhez azonban hozzá kell férnie a felhasználó számítógépéhez. Ahhoz, a spyware-rel történő profilozást a támadó, vagy szolgáltató elvégezze, a spyware-t[18] kell elhelyeznie a felhasználó rendszerében. Ez az esetek igen kis százalékában önkéntes alapon történik, vagyis a felhasználó beleegyezik abba, hogy a szolgáltatás megfigyelje őt, cserébe a szolgáltatás használatáért. Az esetek túlnyomó többségében azonban a szolgáltató a felhasználó tudta nélkül juttatja a programot a rendszerbe és megsérti annak alapvető jogait a privátszférához, amikor merevlemezének teljes tartalmát átkutatja profilépítési céllal.

A legnagyobb probléma ezekkel az alkalmazásokkal, hogy nem csak profilépítésre alkalmasak, hanem mindenhez hozzáférnek, amit a felhasználó a számítógépen tárol, vagy csinál. Ebből kifolyólag a spyware-eket nem elsősorban profilépítésre használják, bár arra is alkalmasak, hanem a felhasználói adatok ellopására és a felhasználó online megszemélyesítésére. Ezen támadások igen nagy számban vannak jelen a mai interneten ezt több statisztika is alátámasztja [19].

2.3.3.5 Cross site scripting (XSS)

A Cross site scripting technológia lényege, hogy a támadó az általa készített futtatható kódot valamilyen módon beágyazza egy másik oldal kódjába, így az ő általa írt szkript a másik oldal kontextusában fog lefutni és hozzáférhet olyan erőforrásokhoz, melyekhez egyébként nem lenne joga. Cross site scripting technológiával például véghez vihet süti lopást [8], vagy kártékony kódot juttathat a felhasználó gépére[9], amely analizálhatja a kliens oldali adatokat.

2.3.3.6 Az automatikusan működő funkciók monitorozása

A felhasználó szokásainak feltérképezésére felhasználhatóak azok a funkciók, melyek a felhasználó számítógépén automatizáltan mennek végbe. Ilyen lehet az RSS, vagy Atom feedek automatizált letöltése, a böngésző kezdőlapjának automatikus betöltődése. Ezek a műveletek mind automatikusan végrehajtódnak, amikor a felhasználó bekapcsolja

---

[18] A spyware egy olyan program, amely monitorozza a felhasználói tevékenységet és az erről készített statisztikákat elküldi az őt üzembe állító személynek, szolgáltatásnak, vagy egyéb módon kémkedik a felhasználó után.

[19] A probléma mértékét igen szemléletesen mutató adatok találhatóak például a Lavasoft honlapján, mely cég elsősorban spyware ls adware eltávolításra szakosodott. A statisztika elérhető az alábbi címen: http://www.lavasoft.com/support/spywareeducationcenter/spyware_statistics.php

számítógépét, vagy elindítja böngészőjét. Ilyen módon a szolgáltatók képet alkothatnak a felhasználó napi rutinjáról, hogy mikor kapcsolja be számítógépét, vagy mennyit tartózkodik előtte, esetleg nyitva van e a böngésző annak ellenére, hogy oldalaikon a felhasználó inaktív.

2.3.4 Védekezés a követésen alapuló profilozás ellen

A követésen alapuló profilozás problémáit látva, ismerve a támadók módszereit a felhasználók számára szükséges volt a védekezés lehetőségének megteremtése. A felhasználó számára rendelkezésre álló eszközkészlet megpróbálja ellehetetleníteni a támadót a felhasználó azonosításában.

2.3.4.1 A böngésző beállításainak módosítása

A legalapvetőbb védekezési mód a böngészőnk beállításainak módosítása olyan módon, hogy az a mi érdekeinket szolgálja, például letiltjuk a sütiket, a külső forrásból érkező objektumok letöltését. Ezek a beállítások azonban nem elegendőek ahhoz, hogy anonimitásunkat biztosítsák, mindössze megnehezítik a támadó dolgát.

2.3.4.2 Anonim proxyk

Az anonim proxyk átjáróként működnek a felhasználó és a szolgáltató között. A szolgáltatóval kizárólag az anonim proxy kommunikál, így a támadó nem juthat hozzá például a mi IP címünkhöz. A proxy esetlegesen bizonyos szűréseket is elvégezhet a tartalmon, eltüntetve az oldalból olyan tartalmi elemeket, melyek lehetővé tennék az azonosításunkat.

2.3.4.3 Anonim böngészők

Az anonim böngészők alkalmazás rétegbeli funkciókkal egészítik ki a proxykat. A visszakövetés elkerülésére alkalmazzák a proxykat, viszont minden esetben alkalmaznak különböző szűréseket és blokkolásokat a tartalmakon, evvel védve a felhasználót. Egy jó anonim böngésző a felhasználóra bízza annak beállítását, hogy milyen információkat osszon meg a szolgáltatókkal és esetlegesen lehetőséget biztosít különböző identitások kezelésére is, többféle, menet közben váltható profil elérhetővé téve.

Az anonimizáló protokollok alkalmazásának azonban komoly korlátai vannak, amikor valaki ilyen megoldásokat kíván használni, figyelembe kell vennie ezeket a tényezőket is.[10]

2.4 A tartalomelemzésen alapuló profilozás

Az eddigiekben a támadó, vagy profilozó elsődleges célja az volt, hogy a felhasználót megfigyelje, nyomon kövesse, és ez alapján próbálja megismerni a szokásait, preferenciáit. Az interaktív internet megjelenésével azonban új lehetőségek nyíltak meg a támadó előtt. A

felhasználók saját maguk állítják elő a profilozáshoz szükséges adatokat, általában publikussá is teszik őket, főleg figyelmetlenségből, vagy nemtörődömségből, a profilépítőnek csak meg kell találnia és elemeznie ezen információkat. A blogok, mikroblogok, fórumok és szociális hálók térhódításával az online szöveges tartalom az önkifejezés ismert és elismert formájává lépett elő, így elegendő mennyiségű információ áll rendelkezésre a profilépítéshez.

Míg az eddigi profilépítés időt és összefogást igényelt, ha a szolgáltatók minél összetettebb, teljesebb profilt szerettek volna, addig az új technológiák mellett a profilépítés megtörténhet úgy is, hogy azt nem előzi meg semmilyen előzetes megfigyelés. A követésen alapuló profilozásnál a felhasználó online műveletei kerülnek rögzítésre, és ha sikeresen összegyűlik a kritikus mennyiség, csak akkor lehetséges belőle stabil, megbízható profil építése. A felhasználó eközben törölheti a sütiket, pitéket, megszabadulhat a spyware-ektől és így az esetleges adatok elveszhetnek. A tartalomelemzésen alapuló módszer használatakor azonban azokat az információkat használja fel a támadó, melyeket a felhasználó maga töltött fel, más szolgáltatóknál összegyűjtött és tárol.

A támadó csak azt akarja kideríteni, hogy a felhasználó vajon mit szeretne, hogy kedvében járva haszonra tehessen szert, hanem, hogy milyen tudásnak, információnak van jelenleg birtokában, milyen a személyisége, hogy az így megszerzett információkat megszemélyesítésre, identitáslopásra, vagy célzott adathalászatra fordíthassa. A támadó a megszerzett adatokkal több módon is visszaélhet, ezeket a későbbiekben részletezzük.

2.4.1 A szolgáltatások

A továbbiakban röviden áttekintjük, hogy milyen lehetőségeket adott a támadó kezébe a WEB 2.0-ás folyamat, megvizsgáljuk, hogy az egyes szolgáltatásokat hogyan fordíthatja a felhasználó privátszférája ellen. Látni fogjuk, hogy az esetek többségében a szolgáltatások nagy mennyiségű információt osztanak meg a felhasználóról publikus módon, így lehetővé téve preferenciáinak megismerését. A szolgáltatásokat, szolgáltatáscsoportokat két nagy kategóriára bontva tárgyaljuk, az először azokat a szolgáltatásokat vizsgáljuk meg, röviden, áttekintő jelleggel, melyekben a megosztott információ jellemzően nem szöveges, majd a szakdolgozat szempontjából fontosabb, nagy mennyiségű szöveges információ megosztására alkalmas szolgáltatások elemzésére kerül sor.

2.4.2 A nem szöveges információn alapuló szolgáltatások

Ezen szolgáltatásoknál a felhasználó tevékenységei nyomán generált meta-adatok, illetve nem szöveges média-tartalom (jellemzően állókép, vagy videó) elemzésével tájékozódhat a támadó. Ezen kategória legnépszerűbb szolgáltatásai jelenleg a videó megosztó oldalak,

online galériák, linkmegosztók, egyesített beléptető rendszerek, hírcsatorna aggregátorok és előszűrők.

Ezen szolgáltatások mind más és más módon engedik be a támadót a magánéletünkbe. Jelen esetben a támadás nem szigorúan informatikai jellegű, a támadó a hétköznapi életünket akarja megismerni, hogy ezen keresztül találjon fogást az életünk informatikai oldalán, a már sokszor említett célzott adathalász levelekkel, megszemélyesítésekkel és identitás-lopásokkal.

2.4.2.1 A videó és képmegosztó szolgáltatások

Ezek a szolgáltatások lehetőséget biztosítnak a felhasználónak, hogy képeit videóit egy interaktív felületen megossza a közösséggel és a barátaival, galériákba, csoportokba szervezze őket, felcímkézze. A közösségnek módjában áll kommentálni, értékelni a tartalmat.

Ezek a képek, videók jó lehetőséget adnak a helyszínek megfigyelésére, feltérképezésére, hiszen a felhasználó sokszor körbejár a helyszínen, néha még a saját lakását is bemutatja.[20] Ezeken a videókon megfigyelhető, hogy az egyes értékes tárgyak a lakásban hol találhatóak, észlelhetőek a riasztó berendezés alkatrészei, vagy akár megjelenhet a felvételen egy, a monitorra ragasztott cédula valamely jelszavunkkal.

Sokszor a felhasználó tudtán kívül is megjelenik online, mondjuk barátai profiljában, így tudtán kívül is építhető róla profil. A felhasználók sokszor nem is gondolják, hogy kikhez juthat el, amit feltöltöttek és ebből sok kellemetlenségük származhat. Ilyen kellemetlenségnek esett áldozatul az a tanárnő is, akinek tánca az internet segítségével bejárta az egész magyar médiát, nagy port kavarva.[21]

Szintén probléma még, hogy ezeket a tartalmakat nem lehetséges csak a szolgáltatónál megtartani, letölthetetlenné tenni, hiszen már a megjelenítéshez is le kell tölteni őket. Ilyen módon az esetlegesen publikussá tett tartalom nagyon könnyen kikerülhet a felhasználó kontrollja alól. A szolgáltató azonban még mindig mindenhez hozzáfér, így ő maga felhasználhatja azt, illetve kompromittálódása esetén semmi sem védi a tartalmat.

---

[20] Ez figyelhető meg például az alábbi videón is: http://www.youtube.com/watch?v=hrSOjcBLBas

[21] A témában megtekinthető egy írás és a videó az alábbi linken:
http://www.blikk.hu/blikk_aktualis/20081030/sokkolta_a_diakokat_a_vetkozo_tanarno/

2.4.2.2 Linkmegosztó és más oldalakat értékelő szolgáltatások

Ezeken az oldalakon a felhasználók linkeket oszthatnak meg egymással, illetve oldalakat osztályozhatnak. Ezekkel a közösségi megoldásokkal igyekeztek az oldal készítői elérni azt, hogy az oldal ténylegesen azokat a linkeket ajánlja, illetve lássa el magas pontszámmal, amelyeket a nagyközönség fontosnak, értékesnek tart. Az ezeken az oldalakon kiemelt tartalmak általában valóban érdekes, releváns információkat tartalmaznak, hiszen ahhoz, hogy magasra kerüljenek az értékelési ranglistán, sokaknak kellett megfelelniük.

A probléma alapvetően az, hogy az egyes ajánlott oldalaknál fel van tüntetve, hogy ki ajánlotta, illetve, hogy kik vették fel még a saját listájukra, így egy adott felhasználó érdeklődési körei feltérképezhetőek a szolgáltatásokon keresztül, hogy mely linkeket mentette el magának, illetve, milyen oldalakat ajánlott a többieknek. Ezek az információk ráadásul előzetes regisztráció nélkül is megtekinthetőek, így a támadó könnyen és kvázi nyomtalanul figyelheti meg a célszemélyt.

A szolgáltatások lehetőséget adhatnak ismerős-hálózat építésére is, hiszen az ismerősök általában figyelik egymás profiljait, megjelölik egymást és a jó oldalakat átveszik, ezért sok lesz a közös linkjük.[11] Mint minden eddigi esetben, most is a szolgáltató az aki az adatok felett korlátlan hatalommal bír és akár igen komoly elemzéseket is végezhet a begyűjtött adatokon, melyek piackutatási célokra igen értékesek lehetnek, hiszen a tömeg véleményét képviselik. Ilyen módon ezek az oldalak a szolgáltatónak és a támadónak is komoly lehetőségeket adnak a kezébe, ha egy, vagy ez esetben akár sok, felhasználó megfigyeléséről van szó.

2.4.2.3 Egyesített beléptető rendszerek

A felhasználók jellemző problémája, hogy ahány szolgáltatást csak igénybe szeretnének venni az interneten, mindenhová regisztrálni kell, mindannyiszor kapnak egy jelszót, melyet vagy fejben tartanak, ami jóformán lehetetlen, vagy felírják őket. Ezt a problémát kívánják a fenti szolgáltatások megoldani. A regisztráció és a felhasználó adatok tárolása már nem egy szolgáltatás része, hanem maga is egy szolgáltatás. A felhasználónak egy fiókja van a fiókszolgáltatónál, a többi szolgáltatást pedig ezen a hozzáférésen keresztül veszi igénybe.

Jelen esetben maga a beléptető rendszer jelenti a kockázatot, személyes adataink szempontjából. Maga a szolgáltatás hatalmas mennyiségű információt kell, hogy tároljon rólunk, hogy az összes többi szolgáltatásnak megfelelő adatokat szolgáltathasson. Egy adathalász szolgáltatással a hozzáférési adatokat megszerezve, ezen információkhoz mind

egy helyen hozzáfér a támadó. Ezen felül természetesen maga a szolgáltató is potenciális profilozóvá léphet elő.

2.4.2.4 Hírforrások, illetve ezeket egyesítő szolgáltatások

A mai nagyobb portálok mind támogatják a hírforrás-kezelést. A technológia lényege, hogy a felhasználónak nem kell ellátogatnia az oldalra, hogy elolvassa az új híreket, hanem egy programban jeleníti meg azokat. A program lehetőséget teremt arra, hogy egyszerre több forrást is figyeljen és jelezze a felhasználónak, ha valahol új információ keletkezett. Így a felhasználó csak betáplálja a hírforrásokat a programnak, a program pedig figyeli azokat és automatikusan letölti az új híreket

Maga az Feed nem jelent problémát privátszféra szempontból egészen addig, amíg a szolgáltatók össze nem fognak, és nem egyesítik a feliratkozott címek listáit, ekkor ugyanis profilokat építhetnek a felhasználókról. Sokkal komolyabb problémát okozhatnak a kiegészítő szolgáltatások. Az online olvasó kliensek összefogják a felhasználó folyamjait, így mindenféle segítség nélkül kapják meg azokat az adatokat, amikhez egyébként a szolgáltatók összefogása kéne.

Még drasztikusabb a helyzet azoknál a szolgáltatásoknál, melyek előszűrést is végeznek, itt ugyanis a felhasználó maga adja meg profilját, hogy az oldal elvégezhesse a szűrést. Vegyük példának a FeedHub üdvözlő oldalának szövegét: *„Give us your feeds. We learn about you. We analyze your feeds to learn about your content preferences. We create a single new feed that contains only the content that best matches your preferences "*[22] – külön érdekesség, hogy az oldal támogatja az OpenID technológiát, így a beállításaink könnyedén összekapcsolhatóak lesznek más szolgáltatásokból származó adatokkal.

Láthatjuk, hogy kizárólag a támadó kreativitása az, ami határt szabhat az információ felhasználásának, hiszen, különösen a képi és videó tartalmak esetén, a médium nem korlátozza a megjeleníthető információkat, így a felhasználók felelőtlenségükben létrehozhatnak olyan tartalmakat, amelyeket egyedi módon lehet felhasználni ellenük, nem a médium, hanem a tartalom sajátosságait kihasználva.

Az itt felsorolt szolgáltatásokon kívül három, igen domináns szolgáltatás uralja az internetet: a blogok, a mikroblogok és a szociális hálók. Ezen szolgáltatások közös jellemzője, hogy az információkat a felhasználó maga adja meg szöveges, tehát könnyen kereshető és értelmezhető formában. A továbbiakban ezek veszélyeit vizsgáljuk meg.

---

[22] A szolgáltatás címe: http://www.feedhub.com/

2.4.3 A szöveges információn alapuló szolgáltatások

Szöveges információn alapuló szolgáltatásként értelmezzük, azokat a szolgáltatásokat, melyekben az információ maga a felhasználói felületen bevitt szöveg, a szolgáltatás ezt tárolja el és ezt jeleníti meg. Ilyen módon a felhasználó közvetlenül generálja a tartalmat, nem pedig csak a tevékenységének következményeként megjelenő információk publikálódnak. (Ilyen lehet például a képeink feltöltése után a galéria létrejötte, vagy egy feltöltött videó, mely konverziókon esik át, mielőtt publikálásra kerülne, vagy link megjelenése a kedvenceink között, miután a „Felveszem a kedvencek közé" gombra kattintottunk.

Ezen szolgáltatások térhódítását elsősorban egyszerűségük, természetességük és könnyű használhatóságuk tette lehetővé. Egy videó feltöltéséhez rendelkeznünk kell egy videokamerával, viszonylag nagy sávszélességgel, ha pedig különösen igényesek vagyunk, akkor videó szerkesztő programokra van szükségünk a tartalom előállításához. A helyzet képek esetében sem sokkal jobb, itt is szükség van az eszközre és az esetleges kiegészítő szoftverekre, a sávszélességen felül.

Szöveges tartalmak esetében azonban a számítógépet leszámítva nincs további eszközigény, sőt, ilyen tartalmak generálásához igen egyszerű és könnyen használható mobil megoldások is léteznek. A felhasználó gyakorlatilag bárhol és bármikor írhat, olvashat szöveges tartalmat, mert sem a személyi számítógépeken, sem a mobil eszközökön nem jelent gondot a szerkesztés és a megtekintés. A tartalom kialakításához, formázásához nem szükséges külső, drága szoftver, mert maga a szolgáltatás nyújt egy intuitív felületet a szerkesztéshez.[23], amely általában nagyon hasonló a megszokott vastagkliens szövegszerkesztőkhöz (MS Word, OpenOffice.org stb), így a felhasználó a tanulási fázist átugorva, egyből minőségi tartalmat produkálhat, formázás szempontjából.

A szöveges tartalmaknak az előállítása nem csak könnyebb és olcsóbb, mint másoké, hanem, az emberi hétköznapokhoz közelebb állva, a szöveges közlés természetességén keresztül gyorsabb is. Sok szolgáltatás, mint például a mikroblogok, vagy a közösségi oldalak üzenő falai, nem a formázhatóság, hanem az extrém egyszerűségen és gyorsaságon keresztül fogják meg a felhasználókat. A mottó: „Bárhonnan, bármikor, bármit, tömören!". A szolgáltatás, minimalitása ellenére lett hihetetlenül népszerű, megragadva a társadalom hatalmas információ szerzési és publikálási vágyát.

[23] Ezek általában kliens oldali weboldalba ágyazható WYSIWYG szerkesztők, melyek kezelőfelülete nagymértékben analóg a megszokott szerkesztőprogramokhoz.

Természetesen az új trendek megjelenése a támadókat, profilozni vágyókat is arra sarkallta, hogy alkalmazkodjanak, és megtalálják lehetőségeiket ezekben a szolgáltatásokban is. A továbbiakban tekintsük át ezt a 3 vezető szolgáltatás típust, hogy milyen veszélyeket rejtenek, elsősorban a szöveges információkon keresztül, de nem szabad elhanyagolni az esetleges mellékszolgáltatásaikat sem, melyek egyéb, néha igen komoly, aggályokat vethetnek fel.

2.4.3.1 Közösségi oldalak

A közösségi oldalakon a felhasználók megadhatják személyes adataikat, feltölthetnek képeket, videókat, ezeket feliratokkal, címkékkel láthatják el, megjelölhetik a rajtuk szereplő személyeket. Klubokhoz csatlakozhatnak, megadhatják korukat, iskoláikat, érdeklődési körüket, nemüket, szexuális orientációjukat. Megadhatják lakcímüket, telefonszámukat, email címüket, MSN-, ICQ-, SKYPE azonosítójukat, gyakorlatilag bármit, függetlenül attól, hogy ez érzékeny, vagy személyes adat-e.

A probléma alapja, hogy az itt felsorolt megadható információk többsége szenzitív személyes adat. Amennyiben a felhasználó nem adja meg megfelelően, hogy ki milyen adataihoz férhet hozzá (Ha a szolgáltatás egyáltalán ad erre lehetőséget!), úgy (szinte) bárki láthatja személyes adataikat, információkat szerezhet arról, hogy a felhasználó mikor hol volt, hol van éppen, vagy hol lesz 2 hét múlva, kiket ismer, mit szeret, milyen politikai, vallási nézeteket vall. Ezeknek az adatoknak a felhasználása igen sokrétű lehet, gyakorlatilag csak a támadó kreativitása szabhat határt.

Megtudhatjuk például, ha valaki elutazik, így otthona őrizetlenül marad. Hobbijának, érdeklődésének megfelelő célzott leveleket küldhetünk neki, melyek adathalász, vagy fertőző oldalakra viszik a felhasználót. Megpróbálhatunk jelszótöréshez személyre szabott szótárat építeni, melyet nyers-erő támadásokhoz használhatunk. Volt gyakorlati példa is arra, amikor valaki mások jelszavait, fiókjait törte fel a megosztott információk alapján.[12]

Különösen érzékeny információ lehet már önmagában is a szexuális orientáció, melyet elsősorban a nem heteroszexuális személyek igyekezhetnek titokban tartani. Ennek ellenére létezik olyan kísérleti megoldás, amely a közösségi portálon bejelölt baráti kör ismeretében jó eséllyel megmondja, hogy az illető homoszexuális e[13].

Pusztán a közösségi oldalak rendeltetésszerű használata is problémákhoz vezethet. Voltak akik azért vesztették el állásukat, mert megosztották főnökükkel, munkájukkal kapcsolatos ellenérzéseiket a közösségi oldalakon.[14] Ezen információk aztán visszajutottak a vezetőséghez és a felhasználó menesztését okozták.

Szintén elgondolkodásra ad okot, az egyes közösségi oldalakon alkalmazott „ismerős-felderítési” lehetőség. A szolgáltatás felajánlja, hogy átnézi e-mail címlistánkat és partnereinket automatikusan felveszi a szociális hálón belüli partnereink közé. Ez az elsőre igen praktikusnak hangzó szolgáltatás, azonban igen sok visszaélésre ad okot. Elsősorban azért, mert a szolgáltató bejelentkezhet az email fiókunkba és nincs garancia arra, hogy kizárólag a címlistánkat fogja átnézni és nem a teljes levelezésünket, vagy naptárunkat és egyéb szolgáltatásunkat, amelyet ugyanavval a hozzáféréssel kezelhetünk. Arra sincs garancia, hogy ebből semmit sem tárol. Ami pedig valós probléma: az ilyen szolgáltatást nyújtó oldalak között található olyan, mely azokra a címekre melyeket nem talál meg a saját adatbázisában, automatikusan meghívót küld, így használva fel címlistánkat saját népszerűsítésére.

További problémákat vet fel az tagging (megjelölés) a feltöltött képek esetében. A felhasználó a feltöltött képen bejelöli a képen szereplőket és megadhatja a nevüket, profiljukat. Ilyen módon valaki akkor is feltérképezheti ismerőseinket, ha mi magunk nem is vagyunk tagjai az adott közösségnek, de ismerőseink bejelöltek minket képeiken.

2-3. ábra - A képen szereplők megjelölése Facebookon

Ezen felül pedig az esetlegesen „zárt”-ként megadott adatokhoz a szolgáltató még mindig hozzáférhet, így fent áll a veszélye, hogy kiadja azokat harmadik félnek, vagy a szolgáltató kompromittálódása révén fér hozzá valaki az egyébként rejtett információkhoz.

2.4.3.2 Blogok, mikroblogok és fórumok

A blog egy igen elterjedt szolgáltatás, mely lehetőséget ad a felhasználónak arra, hogy szöveges tartalmakat, illetve ezekbe beágyazva gyakorlatilag bármit, publikálhassanak. A szolgáltatás hatalmas népszerűsége annak köszönhető, hogy magát a honlapot a felhasználó kézhez kapja, neki csak a kinézettel és a tartalommal kell törődnie. Kézhez kapja a tartalom megírásához szükséges felületet, az értékelhetőséget, kommentlehetőséget, RSS feed támogatást, statisztika készítőket, adminisztrátori eszközöket, hozzáférés vezérlést. Nem kell törődnie a technikai részletekkel, fejlesztésekkel, az a szolgáltató felelőssége.

Különleges alfaja a blogoknak a mikroblog, amely a rövid pillanatszerűen megosztandó tartalmakra koncentrál, bárhonnan bármikor, így lehetőséget ad arra is, hogy internet elérés nélkül, mobiltelefonok segítségével. smsben publikálhassuk a tartalmakat.

A fórumokon egy alapvetően többirányú kommunikáció folyik. Míg a blogok esetében a hangsúly azon van, hogy egy személy publikál egy tartalmat és a többiek esetlegesen hozzászólásaikban kifejtik véleményüket, elképzelésüket a témáról, addig a fórumban egy „feldobott" rövid témaleírás körül aktív beszélgetés, eszmecsere alakul ki, mindenki egyenrangúan vesz részt a kommunikációban.

A problémák itt alapvetően mások, mint a szociális hálózatok esetében. Míg ott a felhasználó maga építette fel profilját és a támadónak csak a közelébe kellett férkőznie, hogy megtekinthesse a teljes profilt. Ha a támadó maga a szolgáltató, akkor ez nyilván elég egyszerű, de egy esetleges külső támadó is könnyen hozzájuthatott a profiladatok egy részéhez, ha a hozzáférés-vezérlés rosszul volt beállítva. Jelen esetben azonban nem egy profil kerül az internetre, hanem adatok, melyek alapján lehetséges a profil előállítása.

A felhasználó személyes, személyiségére, önmagára jellemző tartalmat fog publikálni, ezeket a tartalmakat elemezve lehetséges a profilt felépíteni, vagy egyéb információkat szerezni. A felhasználók beszámolhatnak például arról, hogy éppen milyen értéktárgyakhoz jutottak (pl.: Vettek egy új televíziót[24]), mikor utaznak el, mit fognak venni (ez esetben lehetséges, hogy nagy mennyiségű készpénz áll rendelkezésükre), vagy mi az, amit éppen keresnek. Utóbbi eset például jó lehetőséget ad egy célzott adathalász levél küldésére, melynek témája az a termék lesz, amit a blogger keres.

Különösen veszélyes lehet a mikroblogolás, ahol a felhasználó olyan „jelentéktelen" információt oszt meg mindenkivel, mint, hogy éppen most indult el otthonról, szinte

[24] Rengeteg megfelelő írást találhatunk a Google keresőjét a „i bought a new tv" kifejezéssel használva

kiplakátolja a betörőnek, hogy most gyere, most nem vagyok otthon. (Így tesz például DJ Shashimi is Twitter bejegyzésében[25])

A blogokban az emberek, abban a hiszemben, hogy az csak a célközönséghez jut el, sokszor egészen személyes gondolatokat tesznek közzé, az elmúlt időkben több konfliktus eredt abból, hogy embereket rasszista online megnyilvánulásokkal vádoltak.[26] Amennyiben a tartalmakhoz csak egy szűkebb kör fért volna hozzá, a lelepleződés elkerülhető lett volna. Ezen esetben sem szabad megfeledkezni a szolgáltatói felelősségről, aki még mindig teljes kontrollal rendelkezik az adat felett.

Jól látható tehát, hogy a blogok esetében a veszély elsődleges forrása a blogok és a blogolás alapvető természetéből fakad. Amikor valaki blogot indít, akkor a célja az, hogy gondolatait, véleményét, esetleg érzéseit a nyilvánosság elé tárja és nem gondol bele abba, hogy milyen másodlagos következményekkel járhat ez a tevékenység. Hogyan használhatjuk fel egy felhasználó blogját a privátszférája elleni támadáshoz? Mint már említettük a szöveges tartalmak elemzése alkalmas lehet profilozásra, jelszótörés megsegítésére, illetve megszemélyesítésre is, nagy általánosságban, jelen esetben eltekintve a magából a tartalomból fakadó esetleges egyedi lehetőségekre (mint például a fentebb is említett televízió vásárlás, vagy egy esetleges elutazás konkrét híre).

Profilépítéskor elsősorban az érdekli a támadót, hogy milyen az áldozat, mik a céljai, preferenciái, mit kedvel, mit nem, mire van szüksége, mi az, amivel kapcsolatban kommunikációra, reakcióra lehet bírni. Egy blogot olvasva pontosan ezekhez az információkhoz juthatunk hozzá. Megtudjuk, hogy a szerzőt mi foglalkoztatja éppen, esetleg elolvasva több bejegyzést is, megtudhatjuk, hogy melyek azok a témák, melyekkel a leggyakrabban foglalkozik. Az írások olvasgatása közben rengeteg elejtett információt gyűjthetünk össze, barátok, háziállatok nevét, dátumokat, határidőket, kedvenc filmeket, zenéket, focicsapatokat, rengeteg mindent, ami jellemzi az írót. Ez alapján, esetleg a szociális hálókból szerzett ismeretekkel kiegészítve, már igen komoly és pontos profilt lehet építeni.

Teljesen eltérő felhasználása lehet a szöveges tartalomnak a személyre szabott szótárépítés. A profil, az érdeklődési körök és az előélet ismeretében lehetségessé válik összegyűjteni a felhasználóhoz kapcsolható szavak, számok, kifejezések listáját. Ebből a listából már könnyedén generálható egy olyan komplex szótár, mely egy jelszó elleni szótáras támadásnál

[25] A Google keresőjét az alábbi kifejezéssel használva („i am leaving home site:twitter.com”), rögtön az első találat tökéletesen megfelel a keresésünknek. (http://twitter.com/dj_sashimi/statuses/3931577538)

[26] Egy ilyen jellegű hír tekinthető meg például az alábbi linken a Duna Televízió archívumában: http://www.dunatv.hu/itthon/rasszista_blogger.html

megkönnyítheti a jelszó megfejtését. Az teheti ezt a megoldást különösen hatékonnyá, hogy a kevésbé tudatos felhasználók általában könnyen megjegyezhető jelszavakat választanak és az ember akkor tart könnyen megjegyezhetőnek egy jelszót, ha tudja valamihez kötni, ami az életére, hétköznapjaira jellemző, így a jelszó ezen információk birtokában sérülékennyé válik.

Egy személy jelszavainak megszerzése után előfordulhat, hogy a támadó meg kívánja személyesíteni az illetőt az online környezetben, e-maileket, üzeneteket, bejegyzéseket akarhat írni az áldozat nevében. Általában ezek a megtévesztések annál hatékonyabbak, minél inkább megegyezik a tartalom stílusa és témája az oldalon megszokottakkal. Ahhoz, hogy valakit ilyen módon megszemélyesítsünk, szintén komoly segítséget nyújtat a publikált tartalmak elemzése, bár jelen esetben nem elsősorban tartalom, hanem stílus-analízist kell végezni, jellemző szófordulatokat, mondatokat kell meghatározni, hogy a stílust sikeresen lehessen lemásolni.

2.4.4 Védekezés a tartalom-analízis ellen

A tartalom-analízis alapú profilozás elleni védekezés legalapvetőbb eleme a hozzáférés vezérlés. Amennyiben a támadó nem fér hozzá a tartalomhoz nem képes elemezni azt, így nem képes következtetéseket levonni belőle, nem nyerhet ki információt. A modern szolgáltatások az esetek többségében nyújtanak a felhasználók számára több-kevesebb hozzáférés-vezérlési lehetőséget. Az esetek többségében ez csak a többi felhasználó besorolásának lehetősége „megbízható”/”nem megbízható” kategóriák valamelyikébe és ennek fényben lehet egy tartalmat publikus (bárki láthatja), bizalmas (csak a megbízható felhasználók láthatják), vagy privát (senki sem láthatja) jelzővel ellátva beállítani annak láthatóságát.

Lehet találni azonban olyan szolgáltatásokat is, ahol a tartalmakhoz való hozzáférés egyszerűen, ugyanakkor hatékonyan és változatosan testre szabható, kezdve akár a tartalom láthatóságánál, folytatva a tartalomhoz való hozzászólással, értékeléssel, egészen akár a tartalom szerkesztésének jogáig. Amennyiben a felhasználó tisztában van a tartalom publikálásának kockázatival, akkor ezekkel a beállításokkal már megfelelő mértékben kirekeszthető a külső támadó, amennyiben a szolgáltatás valóban biztonságos, ezekhez az információkhoz kívülről nem lehet hozzáférni.

Ez a megoldás látszólag jónak tűnhet, azonban rengeteg problémát vet fel, felhasználói szempontból. Mindenek előtt a szolgáltató teljes hozzáféréssel rendelkezik a tartalmak felett, így ő maga elemezheti/kiadhatja ezeket az információkat. A szolgáltató esetleges kompromittálódása esetén adataink teljesen védtelenek lesznek. Problémát jelent ezen felül, hogy a hozzáférés vezérlés az egyes szolgáltatások esetén, ha létezik is, akkor sem egységes.

Több szolgáltatás esetén a felhasználónak alkalmazkodnia kell mindegyikhez, több rendszer használatát is meg kell tanulnia.

Ha a felhasználó megfelelő kontrollt szeretne szellemi tulajdona felett, akkor a számára igénybe vehető szolgáltatások listája jelentősen megrövidülhet. A felhasználó arra kényszerül, hogy válogasson a megoldások között, esetlegesen tesztelje azokat, majd kompromisszumos döntéseket hozzon az elérhető szolgáltatások és az információ biztonságos tárolása között, hiszen semmi sem garantálja, hogy a legmegfelelőbb kontrollal rendelkező szolgáltatás az egyéb funkciók terén is kiemelkedően teljesít majd.

2.4.5 A tartalom analízis elleni védelmet szolgáló alkalmazásokkal szemben támasztott alapvető elvárások

A fentiek ismeretében milyen alapvető követelményeknek kell megfelelni egy olyan alkalmazásnak, mely képes a hozzáférés vezérlést olyan módon megvalósítani, hogy az ne függjön a szolgáltatótól (Sem az adat feletti kontroll tekintetében, sem pedig a hozzáférés vezérléshez társított szolgáltatások listájának tekintetében)?

Az elsődleges feltétel a megfelelő hozzáférés vezérlés támogatása. Minél részletesebb, testre szabhatóbb a hozzáférés vezérlés egy szoftver esetében, annál inkább megfelel ennek a feltételnek. Jelen esetben a minimálisan elfogadható funkcionalitás, a legegyszerűbb „igen-nem” hozzáférési modell, vagyis a felhasználó megadhatja, hogy kik azok, akik hozzáférnek a tartalomhoz, és kik azok, akik nem, egyesével döntve, jelszavakat használva, esetleg fekete, vagy fehér listák meghatározásával. A döntés legkisebb egysége nyilván a felhasználó, de a program alkalmat adhat csoportok kezelésére is. Természetesen ezeken felül a szolgáltatáshoz specifikusan alkalmazható szabályok is szükségesek lehetnek.

Míg a hozzáférés-vezérlés megvalósítható akár szerver oldalon is, a mi elsődleges célunk továbbra is az, hogy az adatok felett a kontrollt mi magunk gyakoroljuk kizárólagosan. Tehát a fent összegyűjtött funkciókon felül, még három olyan feltétel van, melynek egy ideális tartalom-analízis megelőző programnak rendelkeznie kell. Az első a kliens oldaliság, a második az univerzalitás és önállóság, a harmadik pedig a kompromisszum-mentesség.

A kliens oldaliság lényege, hogy minden olyan, a tartalmon végzett, művelet, mely a hozzáférés vezérlés megvalósulásához szükséges, történjen meg még a kliensben. Így egyrészt biztosítható, hogy a szabályozások már a szervert is korlátozni fogják a tartalom analízisekor, a szerver is csak akkor láthatja az adatokat, ha megkapta rá az engedélyt, továbbá, hogy ha ugyanazt a hozzáférés vezérlési modellt és alkalmazást szeretnénk több szolgáltatás felett is használni (ld.: univerzalitás), akkor a felület csak így függetleníthető a

szolgáltatástól. Ez a kritérium tehát elsősorban azt szolgálja, hogy az adat feletti kontrollt magunknál tartsuk, vagyis a tartalom addig nem hagyja el a számítógépünket, míg nincs megfelelően védve.

Az univerzalitás lényege, hogy az ilyen módon működő alkalmazás a lehető legtöbb online szolgáltatással legyen képes együttműködni. A cél evvel elsősorban az, hogy a hozzáférés vezérlési felület a lehető legtöbb szolgáltatás felett egységes legyen, így kímélve meg a felhasználót. Hiszen minél erősebb és sokoldalúbb a program, annál fontosabb szemponttá válik az univerzalitás, hiszen előnyeit annál több helyen élvezhetjük.

Az önállóság jelen esetben a szolgáltató függetlenség szinonimája. A program nem lesz képes ellátni a feladatát, ha működéséhez szolgáltatói közreműködés lenne szükséges, de a szolgáltató nem kooperatív, tegyük fel, fent szeretné tartani a feltöltött tartalmak elemzésének jogát. Ha a programunknak szolgáltatófüggetlen, akkor bármely szolgáltatás felett, mellyel képes lenne, technikailag, együttműködni elláthatja a feladatát, akár a szolgáltató ellenére is. Jól látható, hogy az önállóság nagyban növeli az univerzalitást is, hiszen szolgáltatóknak nem kell felkészülniük a program fogadására, illetve nem képesek megakadályozni annak működését.

Az utolsó, és felhasználó szemmel az egyik legfontosabb, szempont a kompromisszum mentesség. Vajon a program használata milyen korlátozásokat jelent a felhasználó számára? Le kell e mondania egyes szolgáltatásokról az adatok feletti kontrollért cserébe, vagy sem? Amennyiben igen, a felhasználói élmény szempontjából mennyire fontos funkciók ezek? Mivel a felhasználók preferenciái igen különbözőek, mind más és más szolgáltatásokat tartanak fontosnak, ezért a fejlesztők számára elsődleges szempont kell hogy legyen a kompromisszum-mentesség. A program fejlesztésekor törekedni kell arra, hogy a program ne okozzon kényelmetlenségeket használat közben.

2.4.6 A meglévő megoldások

Mint már láttuk a hozzáférés vezérlés történhet kliens, vagy szerver oldalon is. A mai online világban elsősorban a szerver oldali, a szolgáltatások által beépítve nyújtott, hozzáférés vezérlések terjedtek el. Mivel ezek igen sokfélék és a dolgozat témája megköveteli, hogy elsősorban olyan megoldásokkal foglalkozzunk, melyek legalább a kliens oldaliság feltételének megfelelnek, ezért a továbbiakban a szerver oldali megoldásokat figyelmen kívül hagyjuk. Megpróbálunk áttekintő képet adni a meglévő kliens oldalon működő szolgáltatásokról és megvizsgálni képességeiket, elsősorban a fent vizsgált szempontok alapján.

A vizsgált megoldások:

1. FaceCloak (http://crysp.uwaterloo.ca/software/facecloak/)
2. NOYB – Social Networks[27]
3. NOYB – Secret Messaging (http://adresearch.mpi-sws.org/noyb.html)
4. FireGPG (http://hu.getfiregpg.org/s/home)

2.4.6.1 FaceCloak

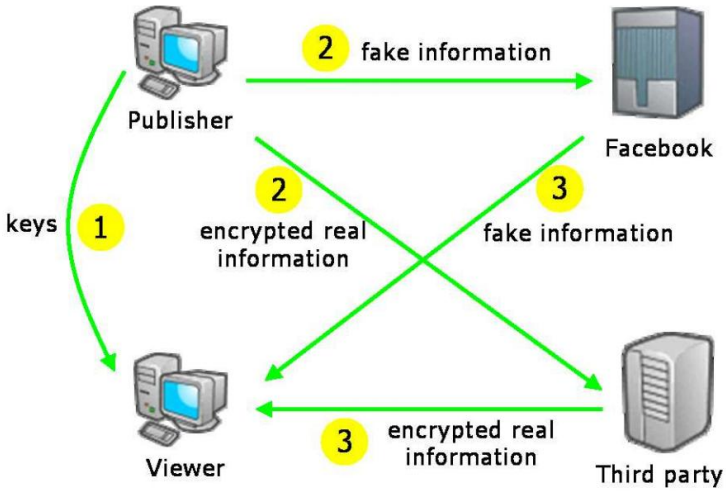
A FaceCloak egy, elsősorban a Facebook közösségi oldalhoz kifejlesztett PET megoldás. A készítői szerint a megoldás bármilyen más közösségi oldalhoz átalakítható, elsősorban popularitása miatt választották kiindulási alapnak ezt a közösségi oldalt. [15]

A program egy Firefox kiegészítésként került implementálásra. Használatához csak ezt a kiegészítést kell telepíteni és a program készen áll a profilinformációk védelmére. A megoldás alkalmazható már meglévő profilokon, de új profilok esetében már a létrehozáskor segítséget nyújt adataink elfedésében. A módszer lényege, hogy a felhasználó rendelkezik egy kulccsal és egy egyedi azonosítóval. Amikor bevisz egy tartalmat az oldalon található űrlapmezők valamelyikébe, és azt megjelöli olyan módon, hogy elé írja a „@@" prefixet, akkor a kiegészítés azon mező értékét kicseréli egy hamis tartalomra, az eredetit pedig titkosítva elküldi egy külső szervernek, egy FaceCloak szervernek, amely a titkosított tartalmat tárolja.

Amikor valaki megtekinti az adatlapot, akkor a hamis adatokat fogja látni. Amennyiben azonban rendelkezik a FaceCloakkal és a hozzáféréshez szükséges adatokkal, akkor a FaceCloak kicseréli a hamis információkat a valósakra a megjelenítéskor. (2-5. ábra) A lényeg tehát, hogy a tényleges tartalmat csak az tudja megtekinteni, aki rendelkezik a hozzáféréshez szükséges információkkal. Se a Facebook, se a külső FaceCloak szerver nem képes az eredeti tartalom visszanyerésére.

A megoldás külön érdekessége, hogy ha egy olyan Facebook felhasználó profillapján kívánunk elhelyezni titkos információkat, aki maga is FaceCloak felhasználó és ismerjük a kulcsát, akkor a tartalom az ő kulcsával tikosítva kerül eltárolása, az ő profiljához csatolva. Ennek oka, hogy a FaceCloak nem a felhasználókat, hanem a profillapokat kezeli, így az egy profillaphoz tartozó információk egyazon kulccsal lesznek titkosítva.

[27] Nem találtam elérhető honlapot, de a működési elv publikusan elérhető [16]



2-4. ábra - A FaceCloak alapvető működése[15]

A fejlesztők a hangsúlyt arra helyezték, hogy a felhasználó részről megkövetelt interakció minimális legyen. Mindössze a kulcsok terjesztését, illetve a titkosítandó tartalmak megjelölését bízzák a felhasználóra. Mint minden olyan alkalmazásnál, mely titkosítással, vagy cserével fedi el a tartalmat, gondolni kell arra, hogy az eredeti szolgáltatás mely funkciói válnak használhatatlanná.

Magától értetődő, hogy jelen esetben azon funkcióktól kell megválnia a felhasználónak, mely érdemben hasznosítaná egy-egy titkosított mező értékét. A Facebook esetében ez elsősorban a felhasználó kereshetőségét befolyásolja, de problémát jelenthetnek a születésnap, névnap emlékeztetők is, hiszen ezek az adatok is lehetnek hamisak. A program legnagyobb hátránya, hogy Internet Explorer alól nem elérhető, valamint, hogy a titkosított profilt egy-egy értelműen hozzárendeli a felhasználó Firefox profiljához, így ha egy Firefox profilt több Facebook felhasználó is használ, akkor a program bevezetése komoly fennakadásokat okozhat. A felhasználó kulcsai online nem hozzáférhetőek, így ha egy távoli számítógépről akarja használni profilját, akkor vagy magával kell vinnie kulcsait, vagy le kell mondania a FaceCloak használatáról.

Univerzalitás tekintetében az szoftver egyértelműen rosszul teljesít, hiszen mindössze egy online szolgáltatással képes együttműködni. Bár a fejlesztők a program ismertetésekor [15] többször is kitérnek arra, hogy az architektúra alkalmassá tehető bármilyen más közösségi oldal védelmére, a program maga erre még sajnos nem képes.

A program az általunk vett értelemben teljesen önálló, semmilyen módosítást nem kell elvégezni a szolgáltatás-szerver oldalán.

Mint minden olyan megoldásnál, ahol a titkosított tartalmat egy kulcs védi és ezt a kulcsot osztjuk szét a felhasználók között fent áll a kulcs-szivárgás veszélye, vagyis hogy kulcsunk valaki olyanhoz is eljut, akivel mi magunk nem szeretnénk azt megosztani. A kulcsok továbbterjesztését semmi sem akadályozza meg.

A szolgáltatás kritikus pontja a külső szerver. Bár a szervernek nem kell feltétlen megbízhatónak lennie, hiszen csak a titkosított információkkal találkozik és az átvitel is már egy titkosított csatornán zajlik, a felhasználó személyes információi kizárólag itt tárolódnak, így ezen szerver elvesztése esetén a felhasználók összes információja megsemmisül, elveszik. A felhasználónak lehetősége van letölteni a szerver forráskódját a project honlapjáról és egy saját szervert üzembe helyezni, ez azonban más problémákat vet fel. A Facecloak egyszerre csak egy szervert képes kezelni, így ha a mi információink egy „A" szerveren tárolódnak, képtelenek leszünk feloldani olyan felhasználók profiljait, melyek nem ezt a szervert használják.

Összesítve tehát a FaceCloak egy közel teljes megoldás a Facebook közösségi oldalon tárolt adatok védelmére, mely a felhasználó számára általában igen kényelmes használatot biztosít. A megoldás érdekessége és bizonyos szempontokból kifogásolható pontja a külső szerver szükségessége, azonban ez garantálja azt, hogy a hamis tartalmak hihetőek legyenek, ugyanakkor helyre lehessen állítani a profiladatokat szükség esetén.

2.4.6.2 None Of Your Business (NOYB) - privacy in online social networks.

A NOYB [16] a FaceCloakhoz hasonlóan egy szociális hálózatok felett működő, általánosnak szánt, de kísérleti jelleggel csak Facebook alá implementált megoldás. A FaceCloakhoz hasonlóan ez is egy Firefox kiegészítés.

Az alkalmazás az előzőekhez hasonlóan behelyettesíti a valós tartalmat egy valósnak tűnővel, azonban a hamis tartalom nem véletlenszerűen generált, hanem egy másik valós felhasználó adata. Az adatok eltárolásához szintén külső tárolókat, úgynevezett szótárakat használ. A felhasználói információk egy szűk halmazát (például a név-nem párost) felhasználva kiszámít egy indexet és a szótárban ennek megfelelő elemmel helyettesíti. A leképezés során egy kulcsot használ, ez hivatott biztosítani a leképezés biztonságát.

A programban a kulcsmenedzsment nem megoldott, a felhasználónak minden egyes lekéréskor be kell ütnie jelszavát. Mivel a felhasználó profilja hamis, bár értelmes adatokkal van feltöltve, így bizonyos alapvető funkciók, mint például a keresés sérülhetnek.

Általánosságban elmondható, hogy a NOYB nagyon hasonló a FaceCloakhoz, hiszen az egyetlen lényeges különbség a két alkalmazás között az ál-tartalom behelyettesítésében nyilvánul meg.

2.4.6.3 None Of Your Business (NOYB) - Posting Secret Messages on the Web

Az előzőekben említett NOYB fejlesztőinek másik terméke. A program arra ad lehetőséget, hogy szöveget titkosítsunk szimmetrikus kulcs alapokon és a titkosított szöveget bárhol publikálhatjuk. Mindenki más aki birtokában van a titkos kulcsnak képes lehet helyreállítani a szöveget.

A program a szöveget, ha kell, egy linkbe rejti, mivel a linkek elküldését a legtöbb online szolgáltatás támogatja, így feltehetőleg nem fog szűrés áldozatásul esni. Természetesen van lehetőség a kiegészítő információktól mentes titkosított szöveg előállítására is. A rendszer tehát néhány egyszerű lépesben használható.

1. Először létre kell hozni egy csoportot és a hozzá tartozó jelszót
2. Titkosítani a tartalmat egy meglévő csoport jelszavával
3. A tartalmat közzé tenni egy weboldalon

A továbbiakban egy felhasználó úgy férhet hozzá a tartalomhoz, ha tagja a csoportnak, vagyis megadtuk neki a szükséges jelszót. Egy titkosított tartalom esetén a felhasználónak ki kell jelölnie azt, majd megadni, hogy mely kulccsal történjen a visszafejtés. Az eddigiekhez hasonlóan ezen program esetében sem megoldott a jelszavak továbbítása, a fejlesztők, valamilyen, a publikálástól független csatornát javasolnak.

A NOYB ezen verziójáról elmondható, hogy teljesen önálló, semmilyen szerver oldali módosítást nem igényel, leszámítva azt, hogy a felhasználónak képesnek kell lennie publikálni.

Ez a verzió abszolút univerzálisnak tekinthető abban az értelemben, hogy a titkosított adat bevitelét a felhasználó egy szövegdobozon keresztül végezheti el, amely a mai internetes oldalak elsődleges beviteli felülete, így majdnem biztos, hogy egy adat titkosított bevitele megvalósítható a NOYB-bal.

A felhasználó részéről azonban fokozott interakciót igényel a program, elsősorban azért, mert az automatizált lehetőségek kiépítése nem valósult meg, hiszen nehezebb lenne a különböző szolgáltatásokhoz alkalmazkodni. Így a titkosítást a felhasználónak önmagának kell elindítania egy, online szolgáltatástól független gomb megnyomásával és a titkosítandó szöveg kijelölésével, valamint a visszafejtés is hasonlóan működik, a szöveg kijelölése után a

visszafejtés gombra kell kattintani. Ezen felül a titkosított szövegen nem hajtódik végre semmilyen szerver oldali művelet, például a BBCode-ok [28] feloldása, így a vizuális élmény károsodik. Ha a szolgáltatások megtanulják felismerni a NOYB által generált karaktersorozatokat, akkor a tartalmak szűrhetővé válnak.

2.4.6.3.1 FireGPG

A FireGPG[29] egy olyan Firefox kiegészítés amely a GNUPG[30] szolgáltatásait teszi elérhetővé a böngészőn belül. A GNUPG az OpenPGP standard (RFC4880) implementációja, ennek megfelelően az alábbi képességekkel rendelkezik, a teljesség igénye nélkül:[31]

- Teljes mértékben helyettesíti a PGPT
- Nem használ szabadalmaztatott algoritmusokat.
- Szűrőprogramként is használható
- Teljes OpenPGP implementáció
- Visszafejti és ellenőrzi a PGP 5, 6 és 7 üzeneteket.
- Támagtva: ElGamal, DSA, RSA, AES, 3DES, Blowfish, Twofish, CAST5, MD5, SHA-1, RIPE-MD-160 és TIGER.
- Könnyen implementálhatóak alá új algoritmusok.
- A User ID-nek kötelező az elfogadott formában lenni.
- Támogatja a kulcs és aláírás lejáratot.
- Több nyelven elérhető.
- Online súgó.
- Opcionális anonim üzenet fogadás.
- Beépített HKP szerver támogatás (wwwkeys.pgp.net).

A Fire PGP segítségével ezek közül a szolgáltatások közül teszi elérhetővé azokat, melyekre böngészés közben szükségünk lehet. Exportálhatunk, importálhatunk kulcsokat, azokat jelszóval védjük. A tartalmakat titkosíthatjuk, aláírhatjuk, akár szimmetrikus, akár aszimmetrikus alapokon. Az esetleges titkosított blokkokat az oldalakban érzékeli, megjelöli és lehetőséget ad a visszafejtésükre. Rendelkezik beépített Gmail [32] támogatással, amely könnyebbé teszi a funkciók használatát Gmail alatt.

---

[28] A BBCode egy korlátozott formázásokat lehetővé tévőleíró nyelv. Elsősorban internetes oldalakon használják, ha az üzemeltető biztonsági okból nem engedélyezi a HTML kódok közvetlen bevitelét a tartalmakba. Ekkor a bevitt kódot az oldal alakítja HTML kódokká.

[29] A szoftver elérhető a http://hu.getfiregpg.org/s/home címen

[30] A GNUPG a http://www.gnupg.org/ címen letölthető

[31] A lista a termék honlapjáról származik

[32] A Gmail a Google online is elérhető levelező szolgáltatása

Összességében tehát a FireGPG egy igen sokoldalú eszköz, amely a PGP technológia használatát igen egyszerűen teszi lehetővé a Firefox felhasználók számára. Felhasználói szöveges tartalmak titkosítására alkalmas, azonban igazi erőssége, az aszimmetrikus titkosítás, nem használható ki azokban az esetekben, amikor egy publikált tartalmat szeretnénk védeni. A program a GNUPG telepítését is igényli, amely egy Firefoxtól független, külső alkalmazás.

Univerzalitását tekintve tehát a FireGPG megfelelően teljesít, bárhol használható, ahol szöveges tartalom publikálására lehetőség van. A program használata szerver oldali módosításokat nem igényel, így akkor is használható, ha a szolgáltatónak nem érdeke, hogy titkosított tartalmat tegyünk közzé.

Felhasználó szempontból a FireGPG talán túl professzionálisnak tekinthető a megcélzott felhasználáshoz. Telepítése viszonylag hosszú, a felhasználónak beállítások hegyein kell magát átvergődnie, valamint telepítenie kell a GNUPG-t is. A program az egyik legnépszerűbb Firefoxba beépíthető FTP klienssel a FireFTP[33]-vel sajnos nem kompatibilis, így a FireFTP használóinak egyelőre választaniuk kell a két alkalmazás között. Ez a program sem fordít különösebb figyelmet arra, hogy a szerver oldali műveltek a titkosított tartalmakon nem mehetnek végbe, így az esetleges formázások (pl.: BBCode) nem lesznek használhatóak a tartalomban.

2.4.7 Az összehasonlítás eredménye

Elmondható tehát, hogy a jelenleg létező megoldások mindegyike képes megoldani a kitűzött feladatot, azonban mindegyikük rendelkezik bizonyos korlátokkal, amelyet elsősorban azok a kompromisszumok okoznak, melyeket egyes más funkciók megvalósítása okozott. Képességeik összehasonlítását az alábbi táblázat tartalmazza.

[33] Az alkalmazás elérhető a http://fireftp.mozdev.org/ címen

	FaceCloak	NOYB *Social Networks*	NOYB *Secret Messaging*	FireGPG
Önállóság	SzI.	SzI.	TÖ.	TÖ.
Univerzalitás	AU., MSzS.	AU., MSzS.	TU.	TU.
Észlelhetőség	NÉ.	NÉ.	KÉ.	KÉ.
Kulcs menedzsment	KM., NKT.	KM., NKT.	KM., NKT.	KM. Támogat PGP kulcsszervereket
Kompromisszumok	MK.	MK.	MK., NA.	MK., FA.
Megjegyzés		Nem találtam működő verziót		OpenPGP*

2-1. táblázat – A programok összehasonlítása

*A program egy külső alkalmazás telepítést igényli, de cserébe kínálja annak teljes funkcionalitását

A táblázatban használt rövidítések magyarázata:

- Önállóság
  - TÖ.: Teljesen önálló
  - SzI.: Saját szervert igényel
- Univerzalitás
  - TU.: Teljesen univerzális
  - AU.: Az architektúra Univerzális
  - MSzS.: Megvalósítás szolgáltatás specifikus
- Észlelhetőség
  - NÉ.: Nehezen észlelhető
  - KÉ.: Könnyen észlelhető
- Kulcs menedzsment
  - KM.: Kulcskezelés megoldott
  - NKT.: Nincs Kulcs Terjesztés
- Kompromisszumok
  - NA.: Nem automatizált
  - FA.: Félig automatizált
  - MK.: Egyes műveletek korlátozottak

Általánosságban kijelenthetjük, hogy a felhasználók számára egy olyan kliens oldali alkalmazás lenne az ideális, mely egyesíti ezen alkalmazások előnyeit és megpróbálja elfedni hátrányaikat.

A programnak elsősorban egyszerűnek, könnyen kezelhetőnek kell lennie felhasználói szempontból azért, hogy tartalmaik publikálásakor a titkosítás folyamata ne legyen számukra kellemetlen, vagy túl bonyolult, mert utóbbi esetekben esetlegesen inkább lemondanának a program használatáról, a kényelem javára.

A programnak a lehető legautomatikusabban kell működnie azért, hogy a felhasználók böngészés közben ne érezzék magukat akadályoztatva. A titkosított tartalmak feloldásának tehát, amennyiben lehetséges, felhasználói interakció nélkül kell végbe menniük.

Szintén az egyszerű és biztonságos használat feltétele, hogy a program ne igényelje se külső alkalmazás, se külső, idegen szerveren tárolt tartalmak elérését. Ezen az áron sajnos elveszik a hihető „áltartalmak” nyújtotta biztonság, azonban egy felhasználótól nem várható el, hogy saját szervert tartson fent, sem pedig az, hogy megbízzon egy idegen külső szerverben.

A titkosításnak erősnek kell lennie, ugyanakkor gyorsan kivitelezhetőnek és alkalmasnak az egy-többirányú közlésre. Ennek eredményeként a legegyszerűbb, ha a program szimmetrikus titkosítást alkalmaz.

A kulcsmenedzsment megoldása ezekben a programokban kritikus, ugyanakkor igen bonyolult feladat, amennyiben a program mellőzni szeretné a külső szerver használatát kulcsmenedzsment célokra, akkor a legpraktikusabbnak az bizonyul, ha a jelszavak továbbítását a felhasználóra bízzuk, figyelmeztetve arra, hogy a jelszót egy független csatornán továbbítsa. Esetlegesen célszerű lehet olyan megoldást biztosítani, ahol több független csatorna együttes információja teszt csak lehetővé a kulcs előállítását.

3 Követelmények egy szöveges tartalom analízis ellen védő technológiával szemben

A fentiek alapján kijelenthetjük, hogy bár a meglévő megoldások mindegyike megoldást jelent a problémára valamilyen mértékben, szükséges megvizsgálni, hogy lehetséges e egy olyan szoftver kifejlesztése, mely egyesíti a fentiek jó tulajdonságit és kompenzálja, megszünteti azokat a problémákat, melyeket használatuk okoz, vagy okozhat esetleg rendelkezik az azokból hiányzó funkciókkal. Elsődleges cél, hogy a program védje a felhasználó privátszféráját, de a program használata nem okozhat kellemetlenséget a felhasználónak. Ennek megfelelően a program használatának egyszerűnek, kényelmesnek kell lennie és az előzőekben felvázolt feltételek közül a kompromisszum mentességnek a lehető legnagyobb mértékben meg kell felelnie.

AZ eddig ismertetett szempontokat, illetve az igényeket ismerve, a programnak az alábbi képességekkel kell rendelkeznie, hogy megfeleljen az elvárásoknak.

3.1 Kliens oldaliság

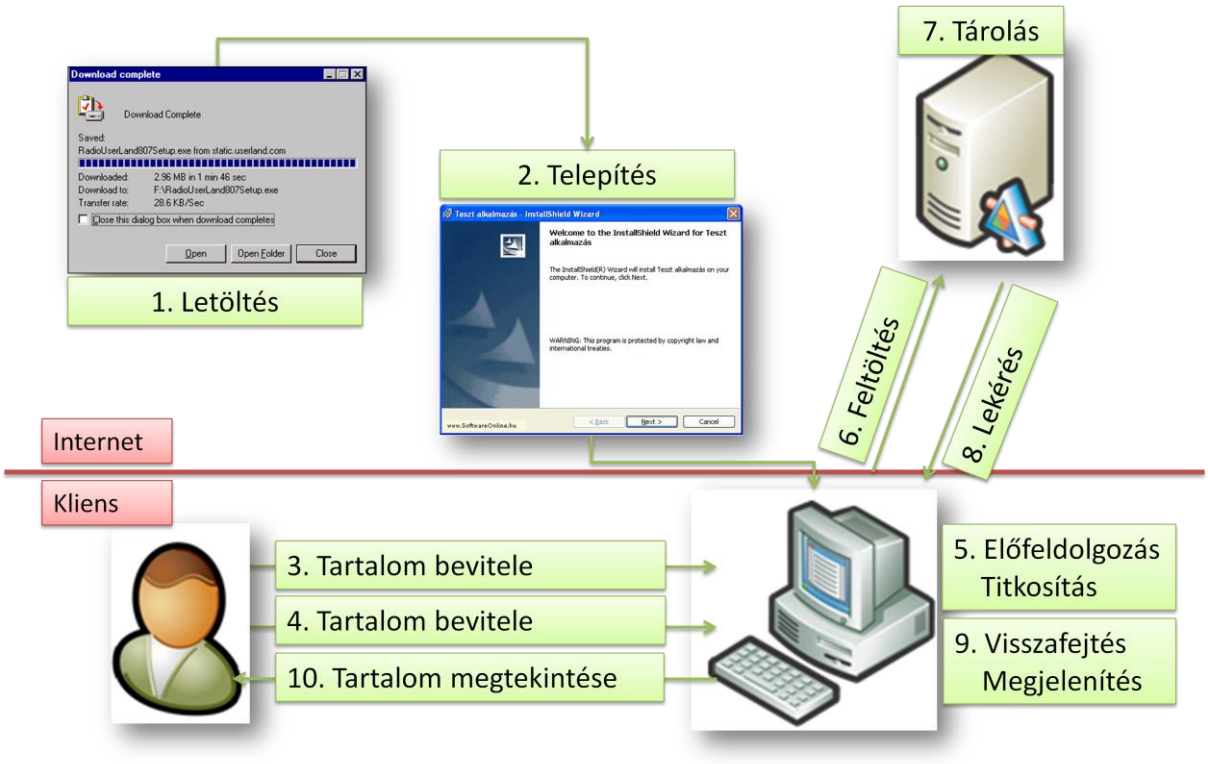
A cél tehát, hogy a szöveg bevitele, előfeldolgozása, titkosítása és utómunkálatai még a szerverre küldés előtt, a kliensben menjenek végbe, a csatornán már az így előállított, védett adat mehessen csak át. A szervernek elsősorban tároló, megjelenítő szerepe van, a titkosított adat a honlap lekérésével együtt letöltődik a kliensbe, visszafejtése már ott történik meg, majd a titkosított tartalom kicserélődik a visszafejtettel, így biztosítva a felhasználói élmény változatlanságát.

Az alkalmazás működése nem igényelhet semmilyen olyan közös tárolót, vagy erőforrást, amely a kliens számítógépén kívül helyezkedik el, így nem alkalmazható semmilyen külső, harmadik fél, vagy önmaga által üzemeltetett szolgáltatás-kiszolgálót igénylő megoldás. Ezek a külső erőforrások ugyanis kritikus pontjai lehetnek a rendszernek, esetleges elveszítésük megbéníthatja a működést, megbízhatóságuknak túl nagy lenne a jelentősége. Nem várható el továbbá egy felhasználótól, hogy megbízzon egy mások által üzemeltetett szerverben, sem pedig az, hogy sajátot tartson fenn.

Ezen felül nem várható el a felhasználótól, hogy a program telepíthetősége más programoktól függjön, mint például a FireGPG-nél, installláskor a programnak az összes szükséges összetevőt tartalmaznia kell. Fontos, hogy az installálás egy egyszerű, intuitív folyamat legyen a felhasználó számára.

A felhasználó számára a beállításoknak egyszerűen áttekinthetőnek és világosnak, mégis kellően részletesnek kell lenniük. Nem szabad túl sok beállítást a felhasználó elé tárni, mert az összezavarhatja, elijesztheti őt a program használatától, ugyanakkor szem előtt kell tartani a megfelelő konfigurálhatóságot is.

Ennek megfelelően a rendszer elvárt működését az alábbi ábra szemlélteti:



3-1. ábra - Az elvárt működés

3.2 Önállóság

A programnak teljesen önállónak kell lennie. A működésnek nem szabad függenie semmilyen külső erőforrástól. A kliens oldaliságnál már említettük, hogy a program nem használhat külső szervert. Az önállóság szempontjából igen fontos, hogy a program mindent tartalmazzon, ami a működéséhez szükséges. Ne kelljen a felhasználónak más alkalmazásokat, idegen forrásból származó könyvtárakat, meghajtókat telepíteni, mert ezek használata esetén fent áll a verziókülönbségekből adódó inkompatibilitás veszélye, vagyis, hogy a tőlünk független külső egység fejlesztői kiadnak egy olyan verziót, mely nem kompatibilis a mienkkel, így a programunk átmenetileg használhatatlanná válhat, amíg mi magunk át nem alakítjuk úgy, hogy megfeleljen az új verzió nyújtotta követelményeknek.

A program működése nem igényelhet semmilyen szerver oldali módosítást, a kiszolgálóval való interakciónak teljes mértékben a szolgáltatás szokásos kezelő felületén, vagy amennyiben a szolgáltatás rendelkezik olyannal, akkor annak hivatalos API-ján keresztül kell megvalósítania a kommunikációt. Mivel a cél az, hogy a szerver tudomása nélkül legyünk képesek a védett tartalom bevitelére ezért kiemelten fontos, hogy csak azokat a felületeket használjuk, melyek egy átlagos felhasználónak is rendelkezésére állnak. Ha szerver oldali módosításokra lenne szükség, akkor csak a szolgáltató előzetes beleegyezése (a szolgáltatás módosítása) után lennénk képesek elérni a funkciókat. A tartalom biztonságának szempontjából igen fontos, hogy a többnyire nem biztonságos, szerver-kliens kommunikációs csatornán, már titkosított adat haladhasson csak át.

Célunk tehát megtartani a szerver kizárólagos tároló szerepét, és elvenni tőle az adat elemzésének lehetőségét.

3.3 Univerzalitás

Minél szélesebb körben alkalmazható az elkészített program, annál hasznosabbnak bizonyul a felhasználók számára. Ezért nagyon fontos, hogy a tervezés és fejlesztés során elsődleges szempont legyen az univerzalitás.

Szöveges információk tekintetében az alábbi szolgáltatás típusokkal és szolgáltatókkal célszerű kompatibilisnek lenni, ha azt akarjuk, hogy programunk a közösség szemében megállja a helyét.

Szolgáltatás	Szolgáltatók/Rendszerek (A teljesség igénye nélkül)
Komplett CMS rendszerek	Drupal, Joomla, PHP-Fusion, PHP-Nuke
Blogok	blog.hu, blogspot.com, freeblog.hu, blog.com, blogger.com
Fórum	sg.hu, PHPBB
Mikroblogok	Twitter.com
Szociális hálók	Facebook, MySpace, Iwiw, MyVIP
Webes levelezők	GMail, Hotmail, Freemail, Citromail, Indamail, Squirrelmail stb.

3-1. táblázat- a jelenleg népszerű szolgáltatások

A szolgáltatók listája a teljesség igénye nélkül készült, azokat a piaci szereplőket tartalmazza, amelyek vagy magyar viszonylatban számítanak jelentősnek, vagy nemzetközileg is használtnak és elismertnek nevezhetőek a magyar tapasztalatok alapján.

Megfigyelve ezeket a szolgáltatásokat kijelenthetjük, hogy a programot két féleképpen lehet felkészíteni az ilyen változatos kihívásokra. Az első eshetőségben a programnak elkészítünk egy magot, amely megát az alapszolgáltatást nyújtja, majd pedig programunkat külön felkészítjük az egyes szolgáltatásokkal való kommunikációra. A módszer előnye, hogy képessé válhatunk az egyes szolgáltatások egyedi jellemzőinek kihasználására, így előfordulhat, hogy valamely szolgáltatás felett sokkal hatékonyabban működhetünk, vagy éppen gazdagabb funkcionalitást tehetünk elérhetővé.

A megoldás ugyanakkor rendelkezik egy hatalmas hátránnyal is. Vegyük észre, hogy igazából nem egy, hanem több programot fejlesztettünk egy közös mag köré, annak minden előnyével és hátrányával. Elsődleges és nem elhanyagolható hátrányként kell kezelnünk ebben az esetben azt a tényt, hogy mivel programunkat specializáltan egy-egy szolgáltatáshoz igazítottuk, ezért a szolgáltatások változásával programunknak alkalmazkodnia kell, ami fejlesztést igényel. Minél több szolgáltatást támogatunk, annál több változást kell tudnunk lekezelni.

A második lehetőség, hogy megkeressük ezeknek a szolgáltatásoknak a közös pontjait, interfészeit, felderítjük azokat a lehetőségeket, amelyek mindegyikben megvannak, és ezekre optimalizálunk. Ennek következményeként csak egy egységes programot készítünk, amelyik képes mindegyik alkalmazással együttműködni. Mivel ezek az interfészek, a szolgáltatások sokszínűségéből adódóan, kellően általánosak, ezért az esetleges változás esélye is kisebb, ha mégis bekövetkezik, akkor pedig várhatóan mindegyik alkalmazást érinteni fogja. Azonban amennyiben valamely szolgáltatás olyan drasztikusan megváltoztatja interfészét, hogy kiesik a kompatibilitási körből, az architektúra nem ad lehetőséget arra, hogy a szolgáltatás a továbbiakban is támogatva legyen, mert a fontos kommunikációs pontokban el fog térni a többitől.

Megvizsgálva a fenti szolgáltatásokat látható, hogy a szolgáltatások közös interfésze az írott szöveg bevitel lehetősége. Szerencsére ezen szolgáltatásoknál a főbb funkciók, illetve maga a szolgáltatás is az így bevitt információn alapul, így ha a titkosítást a szöveg beviteli interfészen alkalmazzuk, akkor pont a szolgáltatások fő információközlő csatornáit sikerül védenünk, így a megoldás a személyes adatok védelme szempontjából igen sikeresnek tekinthető.

Vizsgáljuk meg, hogy amennyiben a szöveges tartalmak titkosítását valósítjuk meg, milyen információkat védhetünk az egyes szolgáltatásoknál.

Szolgáltatás	Fő szolgáltatás	Mellékszolgáltatások
Blogok	Bejegyzés publikálása: Védett	Hozzászólások: védett
Fórum	Hozzászólások beküldése: Védett	Szavazások: a kiírás védett, a numerikus eredmények nem
Mikroblogok	Rövid üzenet publikálása: Védett	Követés: Nem védett
Szociális hálók	Szöveges mezőkben bevitt adatok: Védett	Választásos mezők: Nem védett
Webes levelezők	Szöveges üzenetek cseréje: Védett	Naptár: nem védett
Komplett CMS rendszerek	Ezen rendszerek általában sokféle adatot kezelnek, ezek közül a szövegesek a blogokkal és a fórumokkal azonos szolgáltatásokat nyújtanak, így védettségük is e szerint alakul	

3-2. táblázat - Az egyes szolgáltatások védettsége, szövegtitkosítás mellett

Evvel a módszerrel a vizsgált szolgáltatások fő profiljait minden esetben lefedtük, és csak néhány esetben maradtak védtelen, vagy csak részben védett tartalmak. Levonhatjuk tehát a következtetést, hogy az univerzalitás teljesítéséhez elegendő lehet egy olyan alkalmazás, amely pusztán a szöveges tartalmak titkosítását végzi a szerverre való felküldés előtt. Mint az előzőekben már láttuk, az univerzalitás és a szolgáltatás specifikusság egymásnak részben ellentmondó feltételek, ilyen megvalósítás mellett azonban mégis lehetséges a szolgáltatások kielégítő módon történő védelme.

3.4 Kompromisszum mentesség

A felhasználói szempontból legfontosabb feltétel a kompromisszum mentesség. Az a program, amely a felhasználónak kényelmetlenséget okoz használat közben, nem megfelelő, ugyanis a felhasználó nem lesz vele elégedett, így le fogja törölni az alkalmazást és nem fogja használni. A program tervezésekor oda kell figyelni arra, hogy a program ne korlátozza a felhasználói élményt és a szolgáltatások használhatóságát.

Egy programot akkor értékelhetünk praktikusnak, ha a lehető legminimálisabb mértékben igényli csak a felhasználó közreműködését normál üzem mellett. Ha egy olyan alkalmazást

készítünk, mely bevitt szöveges információt titkosít és felold, akkor a minimális interakció, amire szükség van, az a titkosítandó szöveg kijelölése, a titkosítási kulcs és algoritmus kiválasztása, valamint szükség esetén új kulcs bevitele. Ezt leszámítva mindennek automatizáltan kell működnie. Programunktól tehát elvárhatjuk az alábbiakat:

Titkosításkor:

1. A program adjon lehetőséget a titkosítandó szöveg kijelölésére
2. A titkosítás elindításakor adjon lehetőséget a kulcs és az algoritmus megadására. (Alapértelmezett algoritmus legyen kiválasztva, a minimális interakció a kulcs megadása
3. Jóváhagyás után a program a kijelölt szöveget automatikusan helyettesítse a titkosított megfelelőjével

Az oldal lekérésekor:

1. A program automatikusan észlelje a titkosított blokkokat, oldja fel a titkosítást, helyettesítse be a tartalmat.
2. Amennyiben egy blokkot nem tud feloldani, jelölje azt meg.
3. Amennyiben a felhasználó nem rendelkezik egy kulccsal, legyen lehetősége hozzáadni a listához

A felhasználó szempontjából másik, igen fontos szempont a vizualitás. A titkosítva felvitt tartalom, „tényleges tartalmi részéhez” a szerver nem fér hozzá, így azon nem képes műveleteket végezni. Ezek a műveletek felhasználói szempontból két csoportba sorolhatóak. A vizuális élményt fokozó műveletek (tipikusan a BBCode feldolgozás), valamint a biztonsággal kapcsolatosak, mint például a JavaScript és HTML kódok szűrése. A biztonsági szempont sokkal fontosabbnak tűnhet, azonban vegyük észre, hogy ez a fajta kommunikáció elsősorban olyan emberek között fog zajlani, akik megbíznak egymásban, hiszen a felhasználó privát bejegyzéseinek kulcsát csak az arra érdemesekkel osztja meg, így nem áll fenn a veszélye annak, hogy ilyen módon intézzenek támadást egymás ellen.

A szolgáltatások használhatóságának szempontjából fontos, hogy megkülönböztessük azokat a funkciókat, melyek használhatatlanná válása elkerülhetetlen a program alkalmazásakor. Ezek a funkciók tipikusan a titkosított tartalmon végrehajtható szerver oldali transzformációk és elemzések, mint például a keresés. Nyilvánvaló, hogy a szerver nem fog tudni a titkosított tartalomban olyan keresést végrehajtani, melynek a valós tartalmon kéne végbemenni.

A felhasználót nem érheti kár a szoftver alkalmazásából kifolyólag. Törekedni kell arra is, hogy a szolgáltatás ne tudja megakadályozni a felhasználót a szoftver használatában, meg kell nehezíteni, vagy el kell lehetetleníteni a titkosított blokkok detektálását, már csak azért is, mert egyes szolgáltatások a felhasználók kizárással büntethetik, amennyiben hamis, vagy általuk nem értelmezhető adatokat töltenek fel.

Ezt a funkciót két módon lehet megvalósítani. Vagy szteganográfiával[34][17], vagy pedig a Facecloak és NOYB for Social Networks esetén is alkalmazott, vagy azokhoz hasonló elvű behelyettesítéses megoldásokkal, amikor a titkosított blokknak egy természetes szövegblokkot feleltetünk meg és a felhasználó az általtartalom függvényében juthat hozzá a titkosított blokkhoz. A szteganografikus megoldás teljes értékűnek tekinthető, amennyiben a szöveges tartalom bevitelére szolgáló blokkon nincsenek a tartalom terjedelmére vonatkozó korlátozások, azonban jelen dolgozat témája a titkosításon alapuló megvalósítások vizsgálata, így evvel részletesebben nem foglalkozunk.

A behelyettesítéses technológiák esetében szükséges valamilyen bármikor, bárhonnan elérhető tároló, ahonnan a hamis tartalomhoz kapcsolt titkosított blokk elérhető. Ebben az esetben azonban nem teljesül az a feltétel, hogy a dekódolás során semmilyen külső erőforrást nem használhatunk, így a jelenlegi feltételrendszer mellett ez sem megvalósítható. Jelenlegi megvalósításunk tehát nem lehet képes arra, hogy „hihető” áladatokkal fedje el a titkosítás nyomait.

[34] A szteganográfia célja, hogy olyan módon valósítsa meg az üzenet átvitelét a kommunikáló felek között, hogy a külső szemlélők számára a kommunikáció ténye is rejtve maradjon.

4 Specifikáció

A követelmények figyelembe vétele után jól látható, hogy az elsődleges cél lehet egy olyan maximálisan automatizált alkalmazás készítése, mely képes a bevitt szöveges adatot titkosítani a kliensben, valamint az oldal lekérésekor feloldani a titkosítást. A programnak maximálisan univerzálisnak kell lennie az adott körülmények között, a lehető legtöbb szolgáltatással együtt kell működnie. El kell végeznie a BBCode-ok behelyettesítését és nem kell foglalkoznia hihető „áltartalom" generálással. Ennek megfelelően, a követelmények alapján a programnak az alábbi specifikációs feltételeknek kell megfelelnie.

4.1 Platform

A program Firefox böngésző bővítményként kerül megvalósításra. Ezen döntés eredményeképp a más böngészőket használók (döntően Internet Explorer) ugyan nem lesznek képesek feloldani a titkosításokat, viszont a program fejlesztésekor és karbantartásakor idő és pénz spórolható meg, hiszen csak egy böngészővel kell kompatibilisnek lenni. Ezen felül ez a böngésző valamilyen formában az összes népszerű számítógépes platformon elérhető.

Böngésző kiegészítésként lehetővé válik a program integrálása azon anonim böngészőkbe, melyek Firefox alapúak, így bővítve szolgáltatásaik listáját. Annak bizonyítására, hogy a felállított követelményeknek a program megfelel, vagyis, hogy lehetséges olyan szoftver fejlesztése, mely az elvárt funkciókkal rendelkezik, egy Firefox alapú kiegészítés tökéletesen megfelel, ez a választás kizárólag az implementáció kivitelezését könnyíti meg, abból a szempontból, hogy egyszerűbbé teszi a program és a böngésző kommunikációját.

Amennyiben a program sikereket ér el, és felmerül a széles körű igény a használatára, akkor a fejlesztési tapasztalatok alapján már lehetséges egy Explorer kiegészítés elkészítése is. Az alábbi táblázatban is jól látható azonban, hogy böngészők túlnyomó többségét a Firefox különböző verziói adják, a második jelentős szereplő, az Internet Explorer, viszont verziók szempontjából igen töredezett, a három legutóbbi verzió között közel egyenlő részben oszlanak meg a felhasználók.

2009	IE8	IE7	IE6	Firefox	Chrome	Safari	Opera
Október	12.8%	14.1%	10.6%	47.5%	8.0%	3.8%	2.3%
Szeptember	12.2%	15.3%	12.1%	46.6%	7.1%	3.6%	2.2%
Augusztus	10.6%	15.1%	13.6%	47.4%	7.0%	3.3%	2.1%
Július	9.1%	15.9%	14.4%	47.9%	6.5%	3.3%	2.1%
Június	7.1%	18.7%	14.9%	47.3%	6.0%	3.1%	2.1%
Május	5.2%	21.3%	14.5%	47.7%	5.5%	3.0%	2.2%
Április	3.5%	23.2%	15.4%	47.1%	4.9%	3.0%	2.2%
Március	1.4%	24.9%	17.0%	46.5%	4.2%	3.1%	2.3%
Február	0.8%	25.4%	17.4%	46.4%	4.0%	3.0%	2.2%
Január	0.6%	25.7%	18.5%	45.5%	3.9%	3.0%	2.3%

4-1. táblázat - A böngészők eloszlása[35]

Szintén elmondható, hogy bár a táblázatban nem szerepelnek a Firefox verziók, a jelen probléma szempontjából kizárólag az számít, hogy a Firefoxot használók túlnyomó többsége (a táblázatban található 47%-ból 45%) a böngésző 3.X verzióit használja, amelyek kiegészítés fejlesztés szempontjából gyakorlatilag egyezőnek tekinthetőek. Amennyiben valamely funkció nem érhető el mindegyik verzió számára, akkor az a dokumentációkban minden esetben jelölve van, így megoldható a fejlesztés során, hogy olyan alkalmazás szülessen, amely minden verzióval képes együttműködni.

Az Internet Explorer esetében azonban fő verziószámbeli eltérések vannak, így igen kicsi az esélye annak, hogy egy megoldás mindhárom verzióval kompatibilis legyen. Tehát a Firefox választását az is indokolja, hogy bár így nem tudjuk lefedni a teljes piacot, mégis több felhasználóhoz juthatunk el, mintha bármely másik böngésző-verzió alá fejlesztenénk.

Mivel a Firefox kiegészítések, csak megadott nyelveken fejleszthetőek, ezért a további döntésekkor figyelembe kell venni ezen nyelvek korlátait is. A program logika fejlesztése JavaScript[36] nyelven történik, a felhasználói felület grafikus része pedig XUL[37] nyelven írható le. Ennek megfelelően, amikor valamely funkciót egy harmadik fél által elkészített megoldással készülünk megvalósítani a megoldást ezeken a nyelveken kell keresnünk.

---

[35] forrás: http://www.w3schools.com/browsers/browsers_stats.asp

[36] A JavaScript programozási nyelv egy objektumalapú szkript nyelv, amelyet weblapokon elterjedten használnak. Eredetileg Brendan Eich, a Netscape Communications mérnöke fejlesztette ki; neve először Mocha, majd LiveScript volt, később „JavaScript" nevet kapott, és szintaxisa közelebb került a Sun Microsystems Java programozási nyelvéhez. A JavaScriptet először 1997–99 között szabványosította az ECMA „ECMAScript" néven. A jelenleg is érvényes szabvány az ECMA-262 Edition 3 (1999. december), ami a JavaScript 1.5-nek felel meg. Ez a szabvány egyben ISO szabvány is.

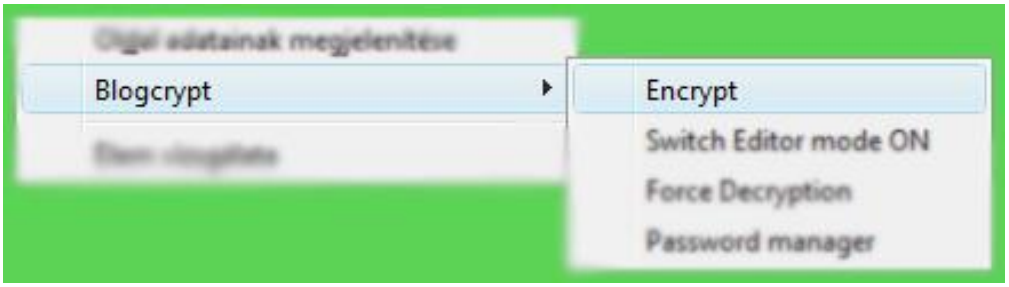
[37] A XUL (XML User Interface Language) egy XML alapú, felhasználói felület készítésére alkalmas jelölőnyelv, amit a Mozilla Alapítvány fejlesztett ki. A Mozilla platformfüggetlen alkalmazásaiban működik, például a Firefoxban. Jelenleg csak a Gecko böngészőmotor tartalmazza a XUL teljes megvalósítását.

4.2 Titkosítási algoritmusok

A program igen fontos eleme a titkosítás. Mivel elsődleges cél, hogy a felhasználónak biztosítsuk a választás lehetőségét, ezért több titkosító algoritmust is alkalmaznunk kell. A probléma méretének racionális keretek között tartása végett a titkosító algoritmusokat külső könyvtár alkalmazásával kell elérhetővé tenni. A választás a PidCrypt[18] könytárcsomagra esett, amely támogatja az AES-CTR és az AES-CBC üzemmódú titkosításokat, valamit rengeteg egyéb, hasznos lehetőséggel bír, támogatja például a BASE64, UTF-8 átkódolásokat, az MD5 és SHA-1/256/384/512 hash megvalósításokat, valamint az RSA titkosítást is.

A titkosítás menete az alábbi módon legyen megvalósítva, hogy a felhasználó számára a lehető legegyszerűbben legyenek elérhetőek a funkciók.

1. A felhasználó begépeli a tartalmat
2. A felhasználó kijelöli a titkosítandó blokkot
3. A környezeti menüből kiválasztja a titkosítás opciót. (erre mutat példát az alábbi. ábra)



4-1. ábra - Encrypt a környzeti menüben

4. Ekkor megjelenik egy ablak, melyben a felhasználó lát egy mintát a kijelölt szövegből, látja az elérhető jelszavak és algoritmusok választékát. A Mégse gombra kattintva megszakíthatja a folyamatot.
5. Az alapértelmezett titkosítási mód az AES-CBC. A felhasználónak lehetősége van más módokat is kiválasztani, jelen verzióban az AES-CTR-t.
6. Alapértelmezett jelszó nincs, azt a felhasználó köteles kiválasztani.
7. Az OK gomb megnyomására, a titkosítás végbemegy. Amennyiben a felhasználó az adatot egy szövegdobozba vitte be, úgy a kijelölt szöveg automatikusan kicserélődik a titkosítottra. Ha a kijelölt szöveg nem olyan helyről származik, ahol a csere megoldható, úgy a titkosított szöveg vágólapra kerül és erről egy üzenetben a program tájékoztatja a felhasználót.
8. A felhasználó a webszolgáltatás segítségével elküldi az üzenetet.

4.3 A titkosított blokk

A titkosított blokk formátuma legyen az alábbi:

[crypt keyid=a_jelszó_azonosítója algo=a_választott_algoritmus_leírója length=a_titkosított_blokk_hossza]

Majd ezt kövesse a titkosított blokk. (A továbbiakban [crypt] tag alatt ezt a struktúrát értem)

Itt látható egy példa a titkosított blokkra: (az átviteli nehézségek elkerülése végett a titkosított blokk BASE64 kódolással kerüljön tárolásra)

[crypt keyid=mykeyid algo=AES-CBC-MD5 length=69] U2FsdGVkX197Gb chyv8Pc7Ryo/RQ ssOmcKFbOvUg+8 dg69eeuJGmMWLg w1Cmd6/C

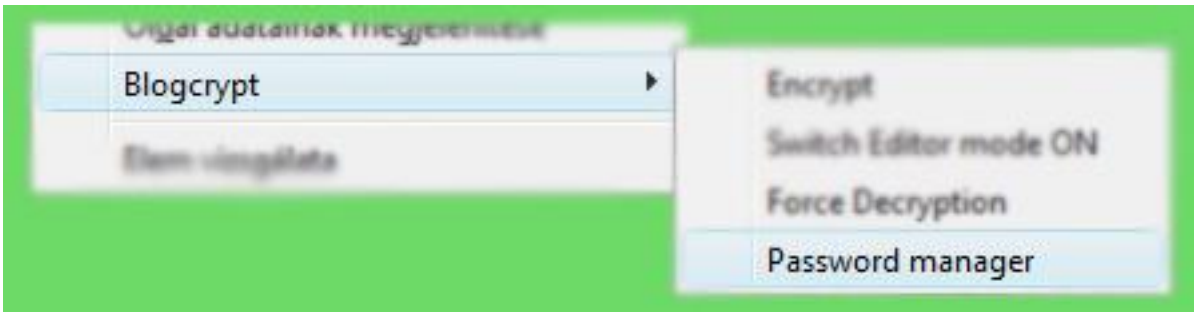
.4-2. ábra - A [crypt] tag struktúrája

A titkosításhoz használt kulcsok az alábbi módon kerüljenek azonosításra és felhasználásra. A kulcsot egyértelműen két paraméter azonosítja. A domain, amelyen alkalmazták, illetve a keyID mező értéke. Ilyen módon elkerülhető, hogy a kulcsok ütközzenek, hiszen egy adott oldalon történő publikálásnál kézben tartható, hogy a keyID mezők ne ütközzenek, míg sok felhasználó esetén globálisan ezt nem lehetne garantálni, előfordulhatna, hogy például két blogon is ugyanazon keyID mellett legyenek titkosítva a tartalmak. A domain alapú elválasztás azonban megszünteti ezeket az ütközéseket.

Ugyanakkor a felhasználónak legyen lehetősége globális érvényű kulcsokat létrehozni, ha a kulcs felvételekor (erről később) a domain mező értékét *„global”*-ra állítja. A globális kulcsok azonban mindig legyenek alárendeltek a domainhez tartozóknak, így ha egy oldalon szerepel globális kulccsal titkosított blokk, de a felhasználó rendelkezik ugyanazon oldalhoz egy domainnel összerendelt kulccsal, akkor a feloldást a program a domain szintű kulccsal kísérelje meg. A globális kulcsok alkalmazhatósága tehát egy fontos szolgáltatás, de a felhasználó felelőssége, hogy az egyedi legyen.

4.4 Jelszó menedzsment

A jelszavak és így a belőlük képzett kulcsok menedzsmentjére szolgáló felület a szintén a környezeti menüből legyen elérhető.



4-3. ábra - A password manager a környezeti menüben

A jelszókezelő felület legyen jól áttekinthető, egyszerű. A felületen a felhasználónak az alábbi lehetőségekkel kell rendelkeznie:

- Jelszó felvétele (a felhasználó kényelme érdekében a domain mező automatikusan kitöltésre kerül az aktív ablak domainjével.
- Jelszó/Jelszavak törlése
- Jelszólista Importálása
- Jelszólista Exportálása
- A jelszólista kiválasztott elemeinek Exportálása

A jelszavak tárolása titkosítva történjen. Létezzen egy mesterjelszó, és a felhasználó csak ezen mesterjelszó birtokában legyen képes hozzáférni a jelszavakhoz, se titkosítani, se feloldani, se hozzáadni, törölni, exportálni ne tudjon ha nem rendelkezik vele. A mesterjelszó bekérése akkor történjen meg, amikor a programnak először szüksége van arra, onnantól pedig a böngésző jelszókezelője kezelje. A böngésző bezárásakor a mesterjelszó kerüljön törlésre a jelszókezelőből. A mesterjelszó kezelésének egyéb módjai is elképzelhetőek, erről részletesebben a további lehetséges fejlesztésekről szóló fejezetben írok.

Az exportáláskor a felhasználó adhasson meg jelszót, mellyel az exportált adatok titkosításra kerülnek. Amennyiben a felhasználó nem ad meg jelszót, úgy a kimenet nyílt szöveg formátumú legyen. Az exportálás eredménye a vágólapra kerüljön, így azonnal beilleszthető, vagy egy fájlba beírva elmenthető, elküldhető.

Importáláskor az adatot a beviteli mezőbe kelljen bemásolni, majd megadni a szükséges jelszót. A program felveszi a listába az új domain-keyid-password hármasokat, amennyiben azonban ütközést észlel, vagyis már létezik az importálandó domain-keyid páros, kérje a felhasználó jóváhagyását ahhoz, hogy a meglévő kulcsot felülírhassa.

4.5 Feloldás

A feloldás történjen teljesen automatikusan. Az oldal betöltődésekor a titkosított blokkokat a program érzékelje, és amennyiben rendelkezik a megfelelő kulccsal oldja fel azt. Ha a feloldás sikertelen, mert a titkosított blokk sérült, vagy nem teljes, akkor a titkosítást jelző blokk előtagot cserélje ki az alábbira:

[Blogcrypt decode error keyID:mykeyid]

Ez előfordulhat például olyankor, amikor a titkosított blokk pont egy bevezető határára esik, így a blokk egy része nem kerül megjelenítésre. Egy ilyen eset megvalósulása és a rá adandó elvárt kimenet látható az alábbi képen:

4-4. ábra - A titkosított blokk egy része lemarad

Ha a felhasználó nem rendelkezik a szükséges jelszóval, akkor a blokk előtag cserélődjön le az alábbira:

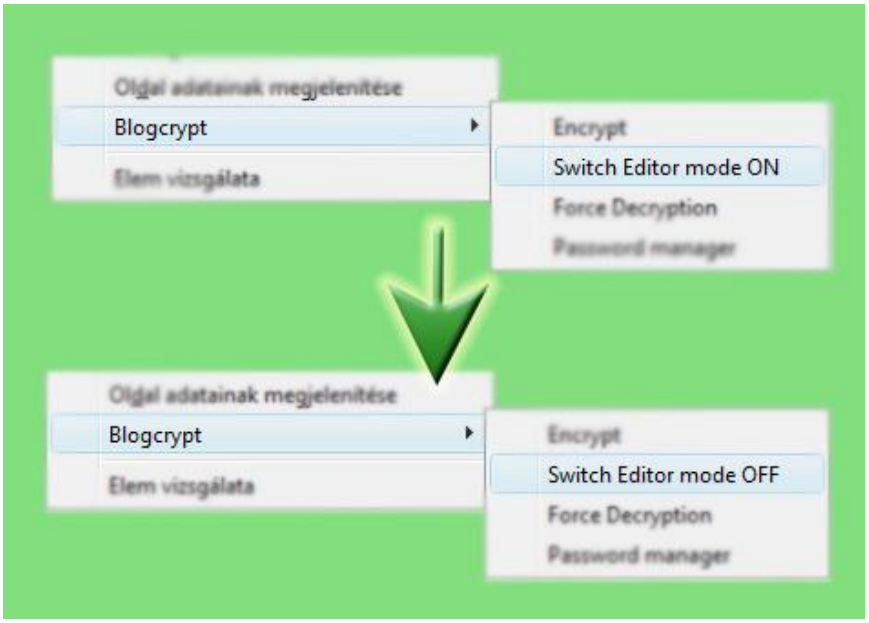
[Blogcrypt password error keyID:mykeyid]

A követelmények alapján a felhasználó ezeket leszámítva semmilyen módon nem értesülhet az esetleges problémákról. Az indok most is az, hogy amennyiben a felhasználót érdekli az a tartalom, amelynek dekódolásakor a hiba megtörtént, akkor észre fogja venni a hibaüzenetet, amennyiben viszont nem érdekli, akkor felesleges megzavarni tevékenységben egy számára nem releváns hibáról történő tájékoztatással.

4.6 Szerkesztő mód

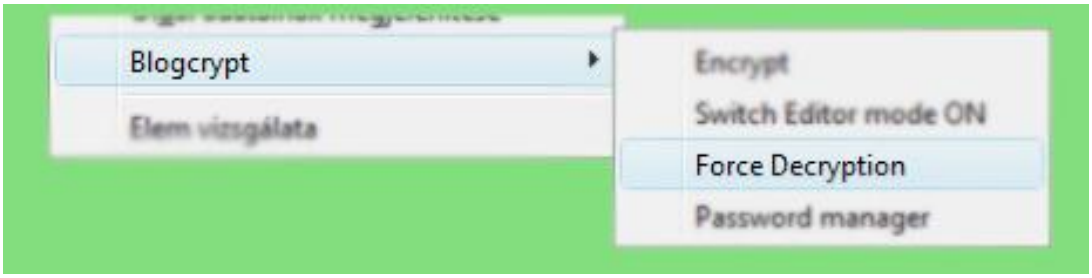
A felhasználónak legyen lehetősége az automatikus feloldás kikapcsolására. Ez a funkció akkor lehet hasznos, amikor a felhasználó szerkesztésre nyit meg egy tartalmat, mely titkosított blokkokat tartalmaz. Ekkor a felhasználó feltehetőleg nem szeretné, ha a már meglévő titkosított blokkok visszafejtődnének, mert akkor az elküldés előtt újra kéne

mindegyiket titkosítani Ezt a funkciót a környezeti menüben lehessen ki-/bekapcsolni, például az ábrán látható módon.



4-5. ábra - A szerkesztő mód ki-/bekapcsolása

Amennyiben a felhasználó szerkesztő módban van, és mégis szeretné, ha az oldal tartalmát a szoftver feldolgozza, akkor legyen lehetősége kézzel elindítani egy visszafejtést az adott oldalra. Az erre szolgáló opció a környezeti menüből legyen elérhető.



4-6. ábra - A kézi indítású visszafejtés a környezeti menüben

4.7 A felhasználó lehetőségei és a felhasználói élmény

A programnak nem szabad a felhasználói élményt jobban korlátoznia, mint az a követelményekben le van fektetve. Ennek értelmében megengedhető, hogy a titkosított tartalmak ne legyenek elfedve, még ha ebből a felhasználónak esetlegesen hátránya is származik, például az üzemeltetők elmarasztalják, mert értelmetlen adatokat tesz közzé.

Ugyancsak elfogadható, hogy a különböző szolgáltatások keresési funkció ne működjenek, hiszen pusztán kliens oldali műveletekkel nem megoldható, hogy a szerver a valós tartalomban keressen.

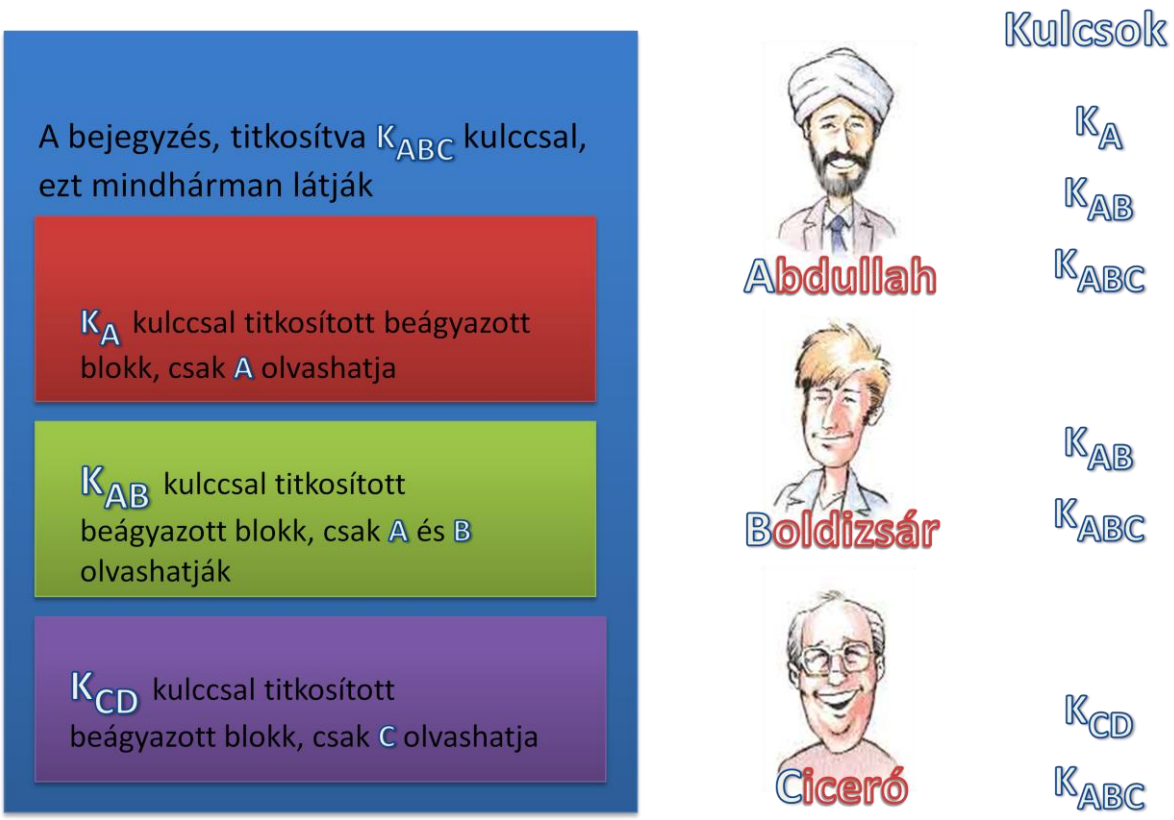
A programnak képesnek kell lennie a BBCodeok feldolgozására, hogy a felhasználó által megadott formázások végbe menjenek a tartalmon. Általános esetben ezt a szerver végezné, azonban mivel az nem fér hozzá a tényleges tartalomhoz, ezért ezt a feladatot át kell venni. A BBCodeok feloldását egy külső forrásból beemelt implementáció végzi[38].

Legyen lehetőség a titkosított blokkok egymásba ágyazására. Ennek a funkciónak a lehetőségeit az alábbi példa illusztrálja.

Az ábrán szereplő példában három ember, Abdullah, Boldizsár és Ciceró olvasnak egy bejegyzést.

- Maga a bejegyzés (kék téglalap) a K_{ABC} kulccsal van titkosítva, ami egy olyan kulcs amivel mindhárman rendelkeznek. Ezért mindhárman el tudják olvasni a bejegyzést.
- A piros blokkot az K_{ABC} kulcson felül védi a K_A kulcs, itt használtuk ki az egymásba ágyazhatóságot, így a piros blokkhoz csak Abdullah fér hozzá.
- A zöld beágyazott blokk az K_{AB} kulccsal van védve, ami Abdullahnak és Boldizsárnak is megvan, ezt jeleztük avval, hogy a kulcs nevében A és B is szerepel, így mind a ketten látják.
- A lila blokkot a K_{CD} kulcs védi, így azt csak Ciceró tudja elolvasni. Vegyük észre, hogy hiába létezne, mondjuk egy Dezső, akinek szintén megvan a K_{CD} kulcs, ő nem férhetne hozzá ehhez a blokkhoz, mert a nagy, mindent magába foglaló kék blokkot nem tudta olvasni, így tudomása sincs erről a blokkról.

[38] A BBCode feldolgozó forrása: http://blogs.stonesteps.ca/showpost.aspx?pid=33



4-7. ábra - Példa az egymásba ágyazásra

5 Tervezés

A tervezés folyamán az egyes komponensek egyesével kerültek megtervezésre, figyelembe véve a követelményeket és a technológia korlátait. Elsődleges célom az volt, hogy a specifikációban nagy részletességgel meghatározott funkciókat és feladatokat modulokba foglaljam úgy, hogy az egy modulba kerülő funkciók valamilyen módon egy logikai egységet alkossanak, illetve tisztázzam a modulok interfészeit, illetve, hogy az egyes funkciók megvalósításáért mely függvények lesznek felelősek, figyelembe véve a modulok közötti esetleges függőségeket is.

5.1 Előkészületek a tervezés megkezdése előtt

Ebben a fázisban az elsődleges cél az volt, hogy megismerkedjek a kiegészítés fejlesztés alapjaival, megtaláljam azokat a forrásokat, dokumentációkat, melyek felhasználásával a későbbi fejlesztés során felmerülő problémákra megoldást találhatok.

5.1.1 Fejlesztési források

A fejlesztés során döntően az alább forrásokat használtam fel. Ezeken az oldalakon példa kódok, dokumentációk, referenciák találhatóak.

- Mozilla Developer[39] : hivatalos Mozilla referencia és példakód oldal, a Mozilla termékekhez történő fejlesztés támogatására
- MozillaZine[40] : Nem hivatalos oldal, ahol példakódok találhatóak
- Javascript Kit[41]: Javascript és Document Object Model referencia oldal

A fejlesztés megkönnyítése érdekében fontos volt, hogy megfelelő fejlesztő környezeteket alkalmazzak mind a Javascript, mind a XUL állományok szerkesztéséhez. A fejlesztőkörnyezetek választásának részleteit a mellékletben tárgyalom.

5.1.2 A Firefox kiegészítés-fejlesztés alapjainak megismerése

Mielőtt a tényleges tervezésbe belekezdhettem volna, megismerkedtem az architektúra lehetőségeivel. Mivel a tervezés során feltétlenül szükséges, hogy a fejlesztő tisztában legyen a lehetőségeivel, ezért néhány online példaprogram, illetve oktató jellegű írás-sorozat elolvasásával és az ott szereplő programok elkészítésével felmértem, hogy mire lehet képes

[39] Mozilla Developer Center: http://developer.mozilla.org

[40] MozillaZine: http://kb.mozillazine.org

[41] JavascriptKit http://www.javascriptkit.com

egy Firefox kiegészítés. Meglévő kiegészítések tanulmányozásával, kihasználva a nyílt forráskód nyújtotta lehetőségeket, igyekeztem olyan forráskód részleteket keresni, melyekkel a későbbi tényleges fejlesztés során felmerülő problémákra találhatok megoldást, illetve ötletet meríthetek belőlük.

5.2 A főprogram és a keret

A kiegészítés összekapcsolását a böngészővel a böngésző felületébe történő integráción keresztül lehet megvalósítani. A fejlesztő elkészít egy XUL állományt, majd meghatározza a XUL állomány kapcsolódási pontját a böngészővel, valamint a XUL állományon keresztül meghatároz legalább egy Javascript állományt, melyet a böngésző szintén betölt.

A kiegészítés minimális működéséhez az alábbi állományok szükségesek.

Állomány	Funkció
valami.xul	Az a XUL állomány, amely a böngészőbe integrálandó új felületi elemeket tartalmazza
valami.js	Az a Javascript állomány, amely a logika kódját tartalmazza.
chrome.manifest	Ez az állomány adja meg, hogy a böngésző felületének mely pontján kapcsolódik a valami.xul.
install.rdf	A kiegészítéssel kapcsolatos adminisztratív információk leírására használatos állomány. Fontosabb mezői:

Az install.rdf fontosabb mezőinek leírását és a megvalósítás során alkalmazott konkrét értékeit a mellékletben részletezem.

Természetesen további XUL és Javascript állományok is készíthetőek, hogy a program könnyebben áttekinthető legyen. Jelen program esetében a belépési pontot a browser.xul állomány képezi. Ennek feladata, hogy tartalmazza a környezeti menüt és annak elemeit, valamint betöltse a Javascript állományokat. Jelen tervezési fázis szempontjából a blogcrypt.js betöltése a fontos, mert az tartalmazza a keretfunkciókat. Ennek az állománynak a feladata az eseménykezelők beállítása, valamint az események kezelése olyan módon, hogy azokat a megfelelő moduloknak továbbítja. A szükséges funkciók és az interfészek összerendelését a melléklet tartalmazza.

5.3 A titkosító funkciók

A titkosító funkciók a Pidcrypt könyvtárat felhasználva kerülnek megvalósításra. A könyvtár használatát illetően példakódok állnak rendelkezésre a Pidcrypt honlapján, ezek alapján végezhető az integráció. A strukturális tagolás végett a titkosítási funkciók mind a crypting_functions.js-be kerülnek. Az itt található funkciók feladata, hogy képesek legyenek a {titkos szöveg, jelszó, titkosítási metódus} hármas birtokában előállítani a nyílt szöveget, illetve a {nyílt szöveg, jelszó, titkosítási metódus} hármas birtokában előállítani a titkos szöveget. A fő- és segédfunkciók és a hozzájuk tartozó interfészek összerendelése megtekinthető a mellékletben.

5.4 A jelszó adatbázis

A jelszó adatbázis célja, hogy a felhasználó jelszavait biztonságosan tárolja a számítógépen. A jelszókezelést megvalósító funkciókat a fileio.js tartalmazza. A jelszó adatbázis titkosítására a beépített titkosító algoritmusok közül, az AES-CBC-MD5 üzemmódot használja, a felhasználó által megadott mesterjelszóval, amelyet a Firefox jelszókezelőjétől kér el. Amennyiben a jelszókezelő nem tartalmazza a mesterjelszót, akkor bekéri azt és gondoskodik a tárolásról. A jelszóadatbázis tárolására a Firefox beépített, beállítás-kezelőjét használja, amiben szöveges formában titkosítva helyezi el az információkat. A szöveges állomány nyílt struktúrája az alábbi legyen:

Egy sor, egy jelszó
Egy soron belül pedig:
domain;keyid;password

5-1 - A jelszavakat tartalmazó szöveges állomány nyílt struktúrája

A szükséges funkciók:

- Jelszó felvétele
- Jelszó törlése
- Jelszó kikérése az adatbázisból
- A teljes adatbázis elkérése

A funkcionalitás megvalósításához szükséges fő- és segédfunkciók és interfészeik összerendelése megtalálható a mellékletben.

5.5 A titkosítás

A titkosítást tartalmazó funkciókat az encryption.js tartalmazza. A modul elsődleges feladata, hogy detektálja az oldalon a kijelölt szöveget, majd titkosítsa azt. Ennek kivitelezéséhez a követelményekben foglaltaknak megfelelően egy ablakot kell megjeleníteni, melyben a felhasználó kiválaszthatja a kulcsot és az algoritmust. Ezt követően titkosítsa a szöveget és cserélje ki a kijelölésben található nyílt szöveget a titkosítottal. Amennyiben a szöveg nem szövegdobozban helyezkedik el, a titkosított blokkot tegye a vágólapra. A titkosítás előtt a nyers szöveg elé fűzi a „blogcrypt" szövegrészt, azért, hogy a visszafejtéskor detektálni lehessen a sikeres visszafejtést. Ekkor a szöveg elején található „blogcrypt" szövegrész hiánytalanul meg kell legyen. A titkosított szöveget lássa el a megfelelő előtaggal és tördelje úgy, hogy 15 karakterenként egy szóközt helyez el benne azért, hogy az egyes oldalakon ne okozzon a szöveg megjelenítése tördelési problémákat. Az ide tartozó funkciók és az interfészeik megtalálhatóak a mellékletben.

A jelszó és algoritmus választót a pswdchooser.xul és a pswdchooser.js alkotja. A felületnek tartalmaznia kell egy listát az elérhető jelszavakról, amelyek közül a felhasználó választhat. Külön csoportosítva jeleníti meg a domainhez tartozó és globális jelszavakat. A felület alapértelmezésben nem jeleníti meg a jelszavakat, de megjelenítésük bekapcsolható. A felhasználónak lehetősége van kiválasztani a titkosítási algoritmust, de ez nem kötelező, az AES-CBC-MD5 mód alapértelmezetten ki van választva. A felület felső részén a felhasználó lát egy mintát a titkosítandó szövegből.

A jelszó és algoritmus-választó grafikus felületének építésekor használt elemek listája megtalálható a mellékletben. Az ehhez tartozó funkciók és a pswdchooser.js-ben történő megvalósítások interfész specifikációi szintén a melléklet részét képezik.

5.6 Az automatikus és kényszerített feloldás

Az automatikus és kényszerített feloldást ugyanaz a modul valósítja meg. Ez a modul a decryption.js állományban található. Az automatikus feloldást az valósítja meg, hogy a keretrendszer az oldal betöltődésekor, az onload eseményre reagálva automatikusan meghívja a feloldó függvényt, míg kényszerített feloldásnál a menüpont hatására fut le. A modulnak képesnek kell lennie megtalálni az oldalban az összes titkosított blokkot, feloldani és kicserélni a titkosított blokkokat a nekik megfelelő nyers szöveggel, illetve hiba esetén kicserélni a [crypt…] előtagot a hibát jelzők valamelyikével. Hiányzó jelszó esetén a [Blogcrypt password error keyID:mykeyid] jelzi a hibát, míg hibás dekódolás esetén a

[Blogcrypt decode error keyID:mykeyid] lesz az új előtag. A decrypt.js összesen egy függvényt tartalmaz, amely ezt a visszafejtő funkcionalitást valósítja meg.

5.7 A jelszó kezelő

A jelszó kezelő elsődleges feladata, hogy egy olyan felületet biztosítson a felhasználónak, melyen képes kezelni jelszavait. A felületnek képesnek kell lennie az új jelszó bevitelére. Az ehhez szükséges mezőkből a domain mező értéke automatikusan kerüljön kitöltésre az aktuális oldal domain-jével. A felület adjon lehetőséget a jelszavak törlésére, importálására és exportálására. Az exportálásnál adott a lehetőség, hogy ne csak a néhány kiválasztott jelszó kerüljön exportálásra. Az exportált adatot a felhasználó jelszóval védheti. Ezt a modult a passwordmanager.xul (grafikus felület) és a passwordmanager.js (logika) valósítja meg.

A passwordmanager.xul felületi elemei és a megvalósításukhoz használható elemek listája megtalálható a mellékletben. A felülethez tartozó funkciókat és interfészeik specifikációját a szintén mellékletben tárgyalom.

5.8 A szerkesztő mód

A szerkesztő mód egyszerűsége miatt nem kapott külön modult. A beállítás a környezeti menüből állítható. Állapotát a böngésző eltárolja, így a böngésző újraindítása esetén is megtartja állapotát. A funkció megvalósítása az onload esemény eseménykezelőjében történik. Amennyiben a szerkesztő mód be van kapcsolva, a feloldási algoritmus nem fut le.

5.9 A tervezés értékelése

A tervezés folyamata során a specifikációban és követelményekben megfogalmazott összes igény kielégítésre került valamely modulban. A tervezés során azon funkciók kerültek egy modulba, melyek logikailag is egy egységbe tartoznak, így biztosítva a kód átláthatóságát és könnyű továbbfejleszthetőségét. A fontosabb szerepű, de gyakran előforduló feladatok külön függvénybe lettek kiemelve azért, hogy a fejlesztés során elkerüljem a redundáns kódolásból adódó veszélyeket.

6 Implementáció és tesztelés

Az implementáció során arra törekedtem, hogy a tervezési fázisban meghatározott függvényeket a meghatározott funkcionalitással megvalósítsam. A modulokat a tervezéskor meghatározott sorrendben implementáltam, hogy elkerüljem a modulok közti függőségekből eredő problémákat, így egy függvény implementálásakor már minden olyan függvény rendelkezésre állt, amelyek más modulokból voltak szükségesek.

6.1 Modulközi tesztelés

Amikor egy modul elkészült, annak teljes funkcionalitását teszteltem, hibák esetén megvizsgáltam együttműködését más modulokkal és lokalizáltam a hibát. A hibát minden esetben javítottam. Amennyiben a hiba egy régebben már késznek nyilvánított modulban volt, és annak módosítása elkerülhetetlen volt, akkor az összes attól függő funkciót újra teszteltem. A fejlesztés közi teszteléseken kívül, a teljes program elkészülte után egy kiterjedt tesztet végeztem, amelyben igyekeztem minél több népszerű szolgáltatással kipróbálni a terméket. A modulközi tesztek eredményeit az egyes modulok implementációs részletei után, a kiterjedt teszt részleteit pedig az implementáció után, külön fejezetben tárgyalom.

Kezdetben a program modulközi tesztelésére egy tesztkörnyezetet állítottam fel, ami egyszerű funkcionalitást valósított meg, képes volt egy szövegdobozba bevitt információt egy gomb megnyomása után megjeleníteni az oldalon a szövegdoboz alatt. Ez a tesztkörnyezet több szempontból is rossz választásnak bizonyult. A legnagyobb problémát az jelentette, hogy az oldal szerkezete és felépítése nem modellezte megfelelően a valós szolgáltatásokat, így az abban megvalósított keresési és visszahelyettesítése algoritmusok a valós szolgáltatásoknál nem működtek.

Ezt a problémát sajnálatos módon csak a feloldó modul implementációja után vettem észre, amikor egy komolyabb tesztelést végeztem. A feloldó modul megléte után a program már minimális ugyan, de képes volt a működésre, így lehetőség nyílt egy köztes külső tesztelésre is. Ekkor váltottam át egy másik tesztoldalra, ami egy egyszerű, de már mások által készített portál volt.[42]. Ezen az oldalon kijavítottam a meglévő modulok alapvető hibáit, majd, mivel tisztában voltam vele, hogy ennek a motornak a szolgáltatásai is jóval elmaradnak a valós helyzetektől, ezért innentől, az immáron nagy mértékben javított szoftvert valós szolgáltatásokon teszteltem.

[42] A teszteléshez a Simple PHPBlog motort alkalmaztam. (http://www.simplephpblog.com/)

Ezek a szolgáltatások a Facebook, a Twitter és a Blog.hu voltak. A végső tesztelésben ezeken kívül egyéb szolgáltatásokat is vizsgáltam. A továbbiakban a modulok tesztelésekor keletkezett eredményeket a fenti három szolgáltatáson végzett tesztek eredményének megfelelően fogom említeni, nem fogok kitérni azokra a hibákra, melyek a rosszul felállított tesztkörnyezetből adódóan keletkeztek, a fejlesztést úgy fogom tárgyalni, mintha egyből ezeken a szolgáltatásokon teszteltem volna, hogy a fejlesztés szempontjából ténylegesen releváns problémák kerüljenek megemlítésre.

6.2 A modulok implementációja

Az alábbiakban az egyes modulok implementációját azok tesztelését és a tesztelések eredményeit ismertetem

6.2.1 A keretrendszer

A keretrendszer fejlesztése a menüelemek overlay.xul fájlban történő elhelyezéséből, illetve a megfelelő eseménykezelők elhelyezéséből állt. Fel kellett vennem eseménykezelőket a kiegészítő betöltődésének kezelésére, valamint a megszűnésére. A betöltődéskor két fontos funkciót kellett megvalósítanom. Elsőként be kellett jegyeznem az automatikus feloldást indító függvényt a weboldalak onload eseményére, valamint inicializálnom kell a beállításokkal a kiegészítést. Ehhez fel kell venni a kapcsolatot a Firefox beállítás kiszolgálójával majd inicializálni a beállításokat, hogy, amennyiben nincs kezdeti értékük, kapjanak egyet. A hivatalos oldalon egy ettől eltérő megoldást ajánlanak a beállítások alapértelmezéseinek megadására, nekem azonban úgy nem sikerült megvalósítanom ezt a funkcionalitást.

Ezt követően a „blogcrypt_editor_mode” beállítás alapján beállítja a szerkesztő módot jelző változót, illetve a menüben található feliratot az ON/OFF értékek közül a megfelelőre. Az automatikus feloldást indító függvényben, ami a weboldal onload eseményére fut le, elhelyeztem egy elágazást, mely nem engedi a feloldás elindítását, amennyiben szerkesztő módban vagyunk. Bár ez a funkcionalitás a legutolsó modult képezte, implementációját azért itt tárgyalom, mert mindösszesen ennyi módosítást igényélt a megvalósítás.

A modul tesztelése során elsősorban azt vizsgáltam, hogy a bekövetkező eseményekre a program a megfelelő függvények hívásával reagál e, illetve ahol minimális logika található (például a beállítások kezelése) ott a logika helyesen működik e. A tesztelés során sok hibát tapasztaltam a beállítások elérésekor, valamint az alapértelmezett értékek beállításakor, abban az esetben, amikor a hivatalos oldalon található módon próbáltam az alapértelmezett értékeket megadni. A hibakeresés során kiderült, hogy az elérés problémáit az okozta, hogy

az alapértelmezett értékek nem jöttek létre, így az alapértelmezés problémájának megoldásával ez a gond is megoldódott. Az alapértelmezett értékek beállítását úgy oldottam meg, hogy betöltődéskor egy függvény lekéri az összes beállítást, és amelyiknél hibára fut, oda beírja az alapértelmezett értéket, mely így a forráskódban tárolódik.

A többi funkciót a hivatalos források alkalmazásával könnyedén sikerült beüzemelni. A keretrendszer implementálása után már megjelentek a menüpontok, a környezeti menüben és a program reagált rájuk.

6-1. ábra - A menüelemek együtt

6.2.2 Titkosító funkciók

A titkosító funkciók implementálása során a Pidcrypt könyvtárait beemeltem a forráskódba és a hivatalos oldalon található példának megfelelően inicializáltam majd meghívtam az ott megadott függvényeket, melyek visszaadták a titkosított blokkot. A Pidcrypt könyvtár nagy előnye, hogy a titkosított adatot automatikusan BASE64 kódolja a könnyebb átvitel érdekében. A két fő funkció a megadott method paraméter alapján a megfelelő segédfüggvény hívják, amely elvégzi a titkosítást.

A modul tesztelése során különösebb problémákat nem tapasztaltam, az integráció sikeresnek bizonyult.

6.2.3 Jelszómenedzsment

A jelszómenedzsment elsődleges feladata a jelszóadatbázis kezelése. A jelszóadatbázis egy titkosított karakterfüzér formájában tárolódik, a beállítás fájlban. Amikor egy jelszót kell visszanyerni az adatbázisból, akkor a beállítás kezelőn keresztül elkéri a titkosított adatbázis. Ezt követően az előző modult alkalmazva visszafejti azt. A visszafejtéshez a mesterjelszót használja, melyet a Firefox jelszó kezelőjétől kér el. Amennyiben az nem tartalmazza a jelszót, akkor bekéri és eltárolja azt. Miután hozzájutott a visszafejtett karakterlánchoz, azt egy mátrixszá transzformálja olyan módon, hogy először sorokra tördeli, majd a sorokat a ’;’ karakterek mentén mezőkre. Az így előállt mátrixban keresi a jelszót, a megadott domain és keyid, alapján. Amennyiben megtalálja, akkor visszaadja azt. Ha nem találta meg, akkor

újabb keresést indít a domain==”global” paraméterrel, globális jelszavakat keresve. Amennyiben így sem talál *false* értékkel tér vissza. Optimalizáció, hogy a domain alapú keresést a sorok felbontásával egyidejűleg végzi.

Amikor egy jelszót kell az adatbázisból törölni, akkor a program elkéri az adatbázist. Ezt az előzőekben is ismertetett módon teszi. Ezt követően előállítja a mátrixot, majd a mátrixon végiglépkedve újra előállítja az adatbázist reprezentáló stringet, azonban a törlendő jelszót kihagyva belőle. Ezt követően eltárolja az adatbázist, mely az alábbi módon működik. Az adatbázis eltárolásához a megkapott stringet titkosítja a mesterjelszóval, majd a beállítás kezelővel elmenti azt.

Egy jelszó felvétele az alábbi módon megy végbe. Az adatbázis a program elkéri, de nem transzformálja mátrixszá. Ezek után az előzőekben említett függvénnyel elkéri az új jelszót az adatbázistól. Amennyiben a jelszóra nem kap találatot, akkor az új jelszót reprezentáló sort az adatbázis végére fűzi, a majd menti azt. Ha találatot kapott, akkor megkérdezi a felhasználót, hogy kívánja e frissíteni a kulcsot. Amennyiben igen, akkor az előzőekben ismertetett függvénnyel kitörli azt az adatbázisból, majd rekurzívan újrahívja magát. A rekurzió azonban ezúttal nem fog találatot kapni a jelszó elkérésére, ezért az első ágba, a jelszó utána fűzését végzőbe lépbe, és felveszi az új, frissebb jelszót.

A modul ezen felül tartalmaz még egy get_passwd_db() nevű segédfüggvényt is. Ez a függvény a későbbi jelszókezelő felületek számára lett beépítve. A függvény előállítja a mátrixot és azt adja át, így ott lehetőséget ad az adatbázis egyszerű feldolgozására.

A komponens tesztelését két fázisban végeztem el. Elsőként úgy, hogy a titkosítási algoritmusok nem lettek alkalmazva, így a jelszó adatbázis nyílt formátumú volt, tehát kézzel szerkeszthető, így könnyedén ellenőrizhettem, hogy a függvények helyesen működnek e. Ezt követően alkalmaztam a titkosítási funkciókat is és így is helyes működést tapasztaltam.

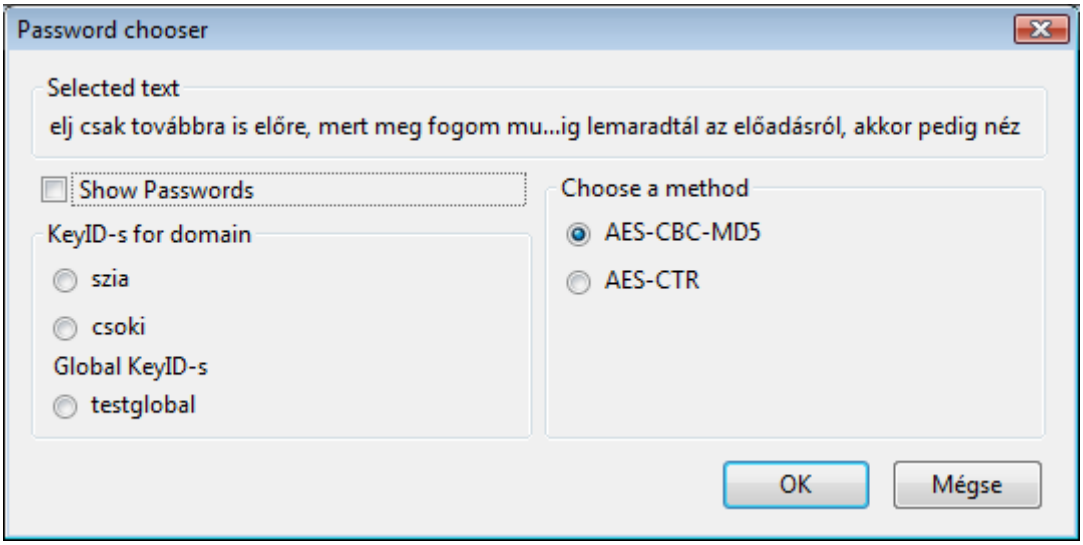
6.2.4 A titkosítás

A program falhasználók által használt két fő funkciója közül ez az egyik, az, amelyik a felhasználó felé a program lényegét a titkosítási funkciót biztosítja. A környezeti menüben található menüpontra kattintva meghívódik a modul függvénye. A titkosítás három lépésből áll.

1. A titkosítandó szövegrész kinyerése
2. A titkosítási paraméterek meghatározása
3. A titkosítás és visszahelyettesítés

Az első szakaszban a böngésző funkcióit kihasználva a program kinyeri az oldalból a titkosítandó szöveget. Miután kinyerte a szöveget, elvégezteti rajta a BBCode-ok feloldását. Ezt követően elindítja a második fázist, megnyitja a jelszó választót.

A jelszó választó felület kilistázza az összes olyan jelszót, mely az adott domainhez tartozik, majd kilistázza azokat, melyek globálisak. Ehhez elkéri az előzőekben tárgyalt modultól a jelszó-mátrixot és maga dolgozza fel azt. Amennyiben a felhasználó bepipálja, hogy az ablak mutassa a jelszavakat is, ne csak a keyID-ket, akkor a program először kitörli az összes elemet a listából, majd újragenerálja azt, ezúttal úgy, hogy a jelszavakat is megjeleníti.



6-2. ábra - A jelszó választó

A program ezen felül kilistázza az elérhető algoritmusokat is. Mind az algoritmus, mind a jelszó választó egy egy radiogroup egységben lett megvalósítva, így garantálva, hogy csak egy jelszó és csak egy módszer legyen kiválasztva. Alapértelmezésben jelszó nincs kiválasztva, ellenben az AES-CBC-MD5 mód az alapértelmezett titkosító. Az OK gombra kattintva, az új, illetve az első fázisból kapott paraméterekkel meghívódik a harmadik fázis.

A harmadik fázisban a program elkéri a titkosítási jelszót, majd titkosítja avval az üzenetet, mely elé odafűzi a „blogcrypt” karaktersorozatot. Ennek funkcióját az előzőekben tárgyaltuk. Az így keletkezett titkosított szövegbe a program 15 karakterenként egy szóközt illeszt. A titkosított blokk elkészülte után a program előállítja az előtagot, belefoglalja a keyid-t, az algoritmust, illetve a titkosított blokk hosszát. Ha a teljes tikos adatblokk elkészült, akkor megpróbálja azt elhelyezni az oldalon, amennyiben ez nem sikerül, a vágólapra helyezi az adatokat és erről tájékoztatja a felhasználót.

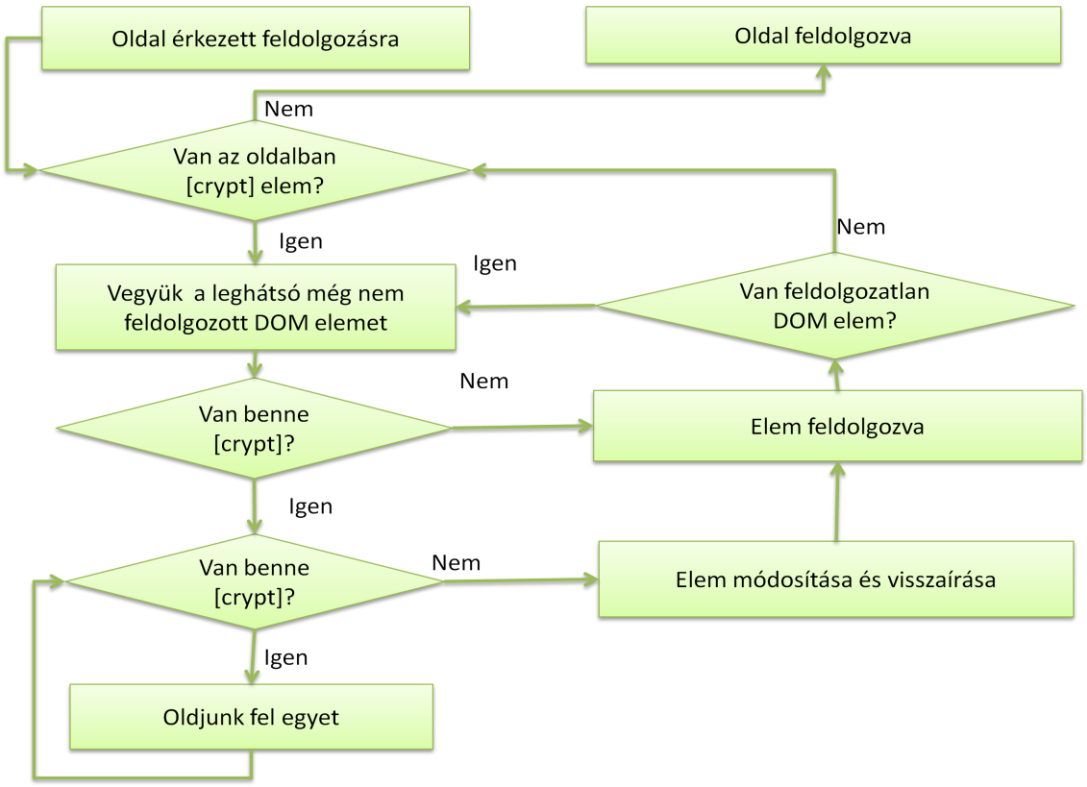
A modul tesztelése során több probléma is akadt, a fent ismertetett megoldás azt az állapotot tükrözi, mely a hibák elhárítása után állt elő. A tesztelés során a titkosított szöveg kinyerése jelentette a legnagyobb problémát, de a jelenlegi implementációban ez a funkció már stabilan működik. Ezen felül problémát jelentett az is, hogy a nagyon hosszú sorok egyes oldalak összeállításánál problémákat okoztak, ennek megoldására vezettem be a titkosított szakasz tördelését.

A program kezdeti verzióiban a titkosított blokk a [crypt] és [/crypt] tagek közé volt zárva és ezt felhasználva történt a titkosított blokk felismerése. A tesztelt oldalak többsége azonban felismerte ezt a keretet és általában különböző intézkedéseket foganatosítottak, hogy az észlelést lehetetlenné tegyék. Ezért vált szükségessé a záró elem elhagyása és a hossz felhasználása. Ez egy másik detekciós problémára is megoldást jelentett, nevezetesen arra, amikor egy oldalon több titkosított blokk volt és az elsőnek lemaradt a záró eleme. Ekkor a program az első blokk nyitó elemét a második záró elemével párosította, ami nyilvánvalóan hibás detekcióhoz vezetett.

6.2.5 Az automatikus és kényszerített feloldás

Az automatikus és kényszerített feloldást ugyanaz az algoritmus végzi, a különbséget az jelenti, hogy az algoritmust az onload esemény elsülése automatikusan, vagy a környezeti menüben való kattintás manuálisan indította el. A modult egyetlen függvény valósítja meg, amely bemeneti paraméterként a feldolgozandó dokumentum referenciáját várja. A feloldó algoritmus folyamatábráját a 7–3. ábra tartalmazza.

Az algoritmus tehát addig folytatja az oldal feldolgozását, amíg van benne [crypt keyid=xxx algo=xxx length=xxx] formátumú tag. A DOM elemek feldolgozását azért hátulról előre valósítottam meg, mert a Javascript *document.body.getElementsByTagName("*");* függvénye a DOM elemeket „bővebb”→ „szűkebb” sorrendben sorolja fel, tehát a sor végén találhatóak azon elemek melyeknek nincs gyermeke. Az algoritmus során elsődleges cél volt, hogy a módosítás az oldal lehető legkisebb egységét érintse csak, ennek okaira a modul tesztelésénél térek ki. A kiválasztott [crypt] tag feloldása a 7-3-as ábrán látható módon történik.



6-3. ábra - A feloldás folyamatábrája

A [crypt] tagek feloldásakor először meghatározom, hogy a DOM elemben hol kezdődik a titkosított blokk, ehhez a [crypt] taghez tartozó reguláris kifejezéssel indítok egy keresést. Ezt követően szintén reguláris kifejezés használatával kinyerem a kódból a taget, majd a három másik reguláris kifejezéssel a paramétereket. Ekkor már birtoklom az összes olyan paramétert, mely a feloldáshoz szükséges. Az előbbiek alapján azt is tudom, hogy a titkosított blokk a DOM elem kódjában mely indexek között helyezkedik el. A visszafejtéshez elkéri a jelszót és kiveszi a felesleges szóközöket a titkos blokkból, majd visszafejteti azt.

Amennyiben a visszafejtés sikeres, a program klónozza az aktuális elemet, majd annak tartalmában kicseréli a titkosított blokkot a feloldás eredményére, majd kiveszi a DOM-ból az eredeti elemet és beszúrja helyére a klónozottat. A feloldás sikerességének ellenőrzésére a titkosításnál a szöveg elé fűzött „blogcrypt” karakterfüzért keresi. Amennyiben a feloldott blokk helyes, akkor evvel a stringgel kezdődik.

A modul tesztelése során egy problémával kellett megküzdenem. A program korai verziói nem használták a DOM elemein történő iterációt, hanem a legkülső DOM elemet a „body”-t dolgozták fel. Ez mindhárom tesztelt szolgáltatás esetén komoly problémákat okozott a megjelenítés során. A problémát az okozta, hogy az oldalakon találhatóak olyan, külső forrásból származó Javascrpitek, melyek létfontosságúak az oldal szempontjából. Ezek azonban az onload esemény kisülése után futnak le. A program tehát az onload eseménykor

az oldal teljes tartalmát kimentette egy változóba, (nyilván ekkor még nem tartalmazta a külső forrás által generált részeket) majd elvégezte a feloldást a kimentett tartalmon és a body teljes tartalmát felülírta.

Mivel a szkriptek nem futottak le a feldolgozásra kimentett adaton ezért a visszaírás után ezek a részek hiányoztak illetve nem működtek, mert felülírtam a lefutott kódot. A hiba keresését nagyban nehezítette, hogy a futási adatok megjelenítéséhez az alert() metódust használtam, ugyanis amikor valamilyen információval felbukkant egy alert(), ami megakasztotta a függvény futását, várakozásra kényszerítve azt, akkor, amíg az OK gombra kattintottam az oldal ténylegesen befejezte a betöltődést, mielőtt manipuláltam volna azt, így ilyenkor a végeredmény helyes volt.

A probléma tehát úgy jelentkezett, hogy normál működés esetén hibákat generált a program, hibakeresés esetén pedig működni kezdett. A hiba okát végül megtaláltam és a problémát megoldottam. A probléma megoldása után a modul megfelelően működött a tesztoldalakon.

6.2.6 A jelszó kezelő

A jelszókezelő egy grafikus felületet valósít meg a jelszavak kezeléséhez. Indulásakor a jelszó adatbázis kezelőtől elkéri a jelszavak mátrixát, majd ezt felhasználva kilistázza az összes jelszót, természetesen a jelszó mező értéket nem megjelenítve. A jelszó mező megjelenítése bekapcsolható, ekkor a program kiüríti a listát, és újragenerálja azt, immáron a jelszavakkal együtt.

A jelszó felvétel és törlés gomb mindössze a jelszómenedzsment megfelelő függvényét hívja meg, a szükséges paraméterezéssel, majd újragenerálja a listát.

Az exportálási funkciók az alábbi módon működnek. Végigiterálnak a kiválasztott jelszavakon, illetve az összesen, ha az lett kiválasztva, felépítik a jelszóadatbázissal azonos struktúrájú stringet(keys), majd veszik annak az md5 hashét(errcorrcode). Ezeket az adatokat összefűzik az alábbi módon:

"<data>\n"+keys+"</data>\n<errcorrcode>"+errcorrcode+"</errcorrcode>"

Az így kapott stringet titkosítja egy, a felhasználó által megadott jelszóval és az AES-CBC-MD5 móddal, majd a vágólapra helyezi. Az ellenőrző hashnek két lényegi szerepe van. Amennyiben a felhasználó nem ad meg jelszót, tehát nyíltszöveges átvitelt választ, akkor ez a hash minimális védelmet jelent a átvitelt illetően. Fontosabb szerepe azonban a helykitöltés, mellyel növeli a titkosítandó szöveg méretét, csökkenti redundanciáját. Ezen felül

szükségesnek tartottam beépíteni azért, mert előfordulhat, hogy a jövőben ide valamilyen egyéb információt lesz szükséges elhelyezni, ekkor ezen a helyen a korábbi verziókkal kompatibilis módon elhelyezhető lesz itt az adat, ugyanis a jelenlegi verzió nem használja fel ezt hibaellenőrzésre. Nyílt szöveges átvitel esetén, ha hiba van, akkor az egyezést kézzel lehet ellenőrizni. Titkos szöveges átvitelnél a dekódolás hibája jelzi az átvitel hibáját.

Importáláskor a megkapott stringet visszafejti, ha szükséges, majd jelszó-mátrixszá transzformálja és a kulcsokat egyesével hozzáadja az adatbázishoz. Az adatbázishoz való hozzáadás megvalósítása garantálja, hogy jelszó csak akkor lesz felülírva, ha a felhasználó jóváhagyja.

A tesztelések során egyetlen érdemleges problémával találkoztam. Az import/export funkciók először az adatokat fájlba írták és onnan olvasták vissza, de az ehhez a hivatalos oldalon ajánlott nsIFilePicker eszköz Linux alatt nem működött, így amíg a problémára nem születik hosszú távú többplatformos megoldás, addig a vágólap használata mellett döntöttem.

6.3 A kész program tesztelése több valós szolgáltatáson

Mivel a modulok a három kiválasztott szolgáltatással működtek, ezért a végső tesztelés során azt vizsgálom, hogy hogyan működik a szoftver a különböző szolgáltatások felett.

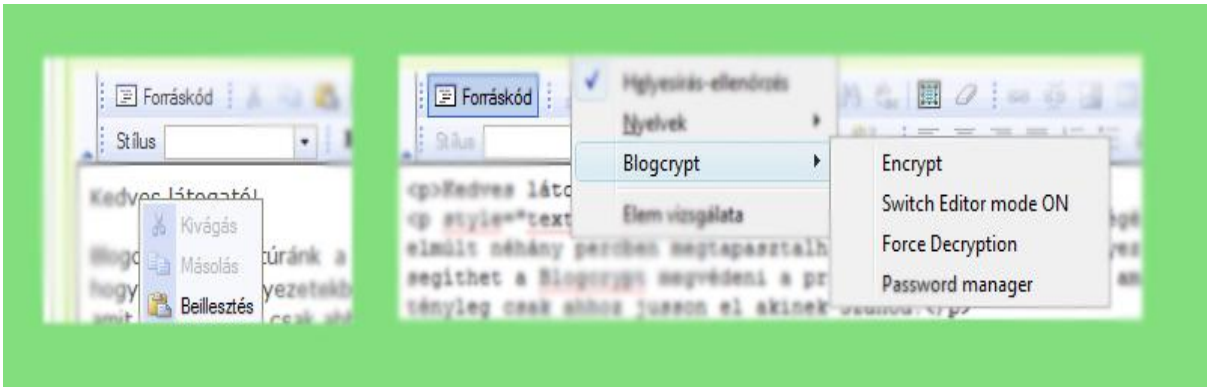
6.3.1 Blog.hu

A program a blog.hu szolgáltatásai felett jól, megbízhatóan működik. Nincs szükség kézzel indított visszafejtés alkalmazására.

A szolgáltatás használata közben két probléma említhető meg, ezek viszont nem oldhatóak meg a kiegészítés keretein belül. Az első kellemetlenség, hogy míg a blog a blogom.blog.hu (például: blogcrypt.blog.hu) címen érhető el, addig a szerkesztő felület a blog.hu domain alatt fut, így az oldal szerkesztőjének a kulcsot redundánsan kell felvennie (vagy globális kulcsot alkalmazni), hiszen a titkosítást csak blog.hu domain alatt tudja véghez vinni, míg a blog olvasása a blogom.blog.hu domainen zajlik. Látható, hogy amennyiben a program nem venné figyelembe az aldomain bejegyzést, akkor a kulcs egyediségének blogon belül tartása sérülne, hiszen az összes blog alá érvényes lenne egy kulcs, amennyiben nincs aldomain ellenőrzés.

A másik probléma a szolgáltatás szerkesztő felületéből származik. A szerkesztéshez egy Javascript alapú WYSIWYG editor tartozik, mely saját környezeti menüt alkalmaz, így nem elérhető a Firefox saját környezeti menüje. A probléma orvosolható, ha a bejegyzés megírása

után a szerkesztőt „forráskód módba” kapcsoljuk, ott ugyanis a funkciók már elérhetőek. Így lehetővé válik a titkosítás.



6-4. ábra - A blog.hu szerkesztő felülete normál és forráskód nézetben

6.3.2 Facebook

Facebook alatt a funkciók megfelelően működnek, a titkosítás lehetséges, nincsenek az előzőekhez hasonló problémák a környezeti menüvel. A Facebook használata során azonban nem kerülhető el a kényszerített feloldás használata, ugyanis az egyes oldalak közötti váltás sokszor úgy valósul meg, hogy a meglévő, már letöltött oldal a szerverhez intézett néhány kérés segítségével újraépíti magát, ekkor azonban az onload esemény nem váltódik ki, így a felhasználó kénytelen a manuális indítást választani. Mivel az oldal igen sok és kiterjedt funkcióval rendelkezik, melyek mindegyikének kimerítő tesztje rengeteg időt és erőforrást igényelne, ezért csak néhány főbb funkció került tesztelésre, ahol a program megfelelően működött.

6.3.3 MyVIP

A program MyVip alatt megfelelően működött, azonban, mivel az oldal itt is sok másodlagos szolgáltatást nyújt, nem volt lehetőség mindet kimerítően tesztelni, csak a főbb funkciókra hagyatkoztunk.

6.3.4 Iwiw

Az Iwiw.hu alatt a program szolgáltatásai kifogástalanul működnek, kizárólag az üzenő fal olvasásakor szükséges a kényszerített feloldás, mert az a Facebookhoz hasonlóan dinamikusan építi fel önmagát, így az onload érvényre jutásakor még nem része az oldalnak.

6.3.5 Twitter

A Twitterrel a program kifogástalanul együttműködik. Mindösszesen egy új üzenet felvételekor következik be az az esemény, mint a Facebook és Iwiw üzenő falainál, ugyanis a

legújabb üzenet oldalbetöltődés nélkül kerül az üzenetsor tetejére a küldő számára, így ő a legfelső üzenetet csak az oldal újratöltése, vagy a kényszerített ellenőrzés után olvashatja. Tekintettel azonban arra, hogy ő maga írta ezért ismeri a tartalmát, ez nem tekinthető igazán komoly problémának.

Sokkal komolyabb problémának tekinthető, hogy titkosított blokk mérete nagyságrendekkel nagyobb, mint az eredeti szövegé, a titkosító alkalmazásával a 140 karakteres üzenethossz a töredékére csökken. Twitteren a rövid üzenethossz korlátai miatt amúgy is elterjedt, hogy a felhasználók csak linkrövidítőkkel[43] lerövidített linkeket küldenek egymásnak, ezen rövidített linkek azonban kényelmesen elférnek kevesebb karakteren is, így ez a felhasználások egy részében nem jelent komoly korlátot.

6.3.6 Gmail és Hotmail

Bár a program alapvetően nem ezen szolgáltatásokhoz lett fejlesztve megéri megvizsgálni, hogy milyen mértékben válik igazzá az a követelményekben lefektetett elmélet, mely a program univerzalitására vonatkozott. A követelményekben elsősorban a bevitel univerzalitása szerepelt és ennek megfelelően mindkét szolgáltatással képes is a program együttműködni. Igaz az automatikus behelyettesítés nem valósul meg, de a szöveg vágólapra kerül, így lehetséges megírni a titkosított üzenetet. Feloldáskor azonban mindkét szolgáltatás megakadályozza a titkosított blokk megtalálását és helyettesítését, így a kényszerített feloldás használata szükséges.

A Gmail lehetőséget ad tiszta HTML alapú nézet használatára is, ekkor a program kifogástalanul működik.

6.4 A tesztelés értékelése

A tesztelés során megállapítottuk, hogy program helyesen és stabilan működik, azokban a környezetekben, melyekben teszteltük. Igyekeztünk a fontosabb szolgáltatásokra helyezni a hangsúlyt, hogy a szoftver a lehető legszélesebb körben alkalmazható legyen. A továbbiakban összefoglalom a fejlesztés tapasztalatait, majd pedig megvizsgálom, hogy a program mennyire felel meg a követelményekben támasztott négy alapfeltételnek.

[43] A linkrövidítő szolgáltatások a felhasználók által megadott linkek helyett a felhasználó kezébe adnak egy rövidebb linket. Amikor a felhasználó meg akarja nyitni a rövidített linket, akkor a linkrövidítő átirányítja őt a valós linkre.

6.4.1 A fejlesztés tapasztalatai

Tekintettel arra, hogy a fejlesztés kezdetén még semmilyen tapasztalattal nem rendelkeztem Firefox kiegészítések fejlesztése terén, illetve XUL ismeretekkel sem rendelkeztem, Javascript ismereteim is igen hiányosak voltak, ezért a fejlesztés során igen nagy mennyiségű új tapasztalatra tettem szert. Ezek egy része magukhoz a konkrét nyelvekhez, illetve a Firefox kiegészítők felépítéséhez kapcsolódik, más részük azonban sokkal általánosabb ismeretként fogalmazható meg.

Általánosságban elmondható, hogy egy közelítő képre tehettem szert a Firefox kiegészítések lehetőségeit illetően. A fejlesztők rendelkezésére álló eszközök és interfészek rengeteg mindenre képessé tehetnek egy kiegészítést, azonban pontosan a funkciók nagy mennyisége az, ami nehezen kezelhetővé teszi a problémák megoldását a tapasztalatlan fejlesztő számára. Többször szembesültem avval, hogy azért okozott nehézségeket egy probléma megoldása, mert nem voltam tisztában a lehetőségeimmel, illetve a rendszer képességeivel.

Ugyanakkor látom, hogy mi mindent lehetséges Firefox kiegészítésként megvalósítani és elmondható, hogy az architektúra nem szab komoly korlátokat a fejlesztő számára, amennyiben a fejlesztő már tisztában van a lehetőségekkel.

Komoly probléma azonban, hogy jelenleg nincs egy komplex, koherens fejlesztő környezet, amely a fejlesztést segítené, különösen hiányoznak az automatikus kiegészítés funkciói. Ezek a funkciók a kezdő fejlesztők számára jelentenek hatalmas segítséget, mert egy adott helyzetben képesek felvázolni a lehetőségeket, így segítve a továbblépést. Nagyon sokat segítenék a fejlesztést a hibakereső funkciók, mint például a lépésenkénti végrehajtás.

A fejlesztés másik fontos tapasztalata olyan Firefox kiegészítés fejlesztése során mely a weboldalakat manipulálja elengedhetetlen a HTML szabvány és a DOM pontos ismerete, valamint igen nagy előny, amennyiben a fejlesztő tisztában van a jelenleg élen járó szolgáltatásoknál alkalmazott technológiákkal, így képessé válhat hozzájuk alkalmazkodni. A program HTML-t manipuláló részeinél én magam is sokszor szembesültem azokkal a hátrányokkal, melyeket utóbbi ismereteim korlátozottsága okozott. Az egyes részeket akár többször is át kellett írnom, hogy egy olyan megoldást találjak, mely mindegyik szolgáltatással kompatibilis.

Szintén elmondható, hogy a szerver oldali programozásban szerzett tapasztalatok nagy előnyt jelenthetnek egy ilyen alkalmazás fejlesztésénél, hiszen amennyiben az ember fel tudja mérni a szerver oldal képességeit és korlátait, amihez tapasztalat szükséges, akkor szintén könnyebben tud alkalmazkodni ahhoz. Erre jó példa, hogy a kezdeti struktúrát, a

[crypt], [/crypt] tagek alkalmazását a szolgáltatások többsége kiszűrte és valamilyen módon tett ellene. Jól jellemzi az internet és a probléma sokszínűségét, hogy nem találtam két olyan szolgáltatást a tesztelés közben, melyek azonos módon oldották volna meg a struktúra megbontását. Volt szolgáltatás, amely szóközt injektált, volt, amelyik karaktereket, de volt olyan is, amely egyszerűen csak levágta a karaktersorozat végét.

Mindezeken felül azonban elmondható, hogy a szolgáltatások többsége nincs megfelelően felkészülve az ilyen jellegű „támadásra". Szolgáltatói oldalt lehetetlen felkészíteni arra, hogy az összes ilyen próbálkozást szűrni tudja, ellenben a leggyakoribb típusokra a szolgáltatónak már érdeke megoldást fejleszteni. Amíg egy ilyen, a szolgáltató ellen indított „támadás" a szolgáltató mércéjével mérve nem jelentős, addig a program használóinak nem kell aggódniuk amiatt, hogy a szolgáltató esetlegesen retorziókat alkalmaz.

6.4.2 Megfelelés a követelményeknek

Az alábbiakban megvizsgáljuk, hogy a program mennyire felel meg a követelményekben és a specifikációban foglaltaknak. A kliens oldalisággal és az önállósággal ellentétben, melyeknek egyszerű és ésszerű szabályok bevezetésével meg lehet felelni, az univerzalitás és a kompromisszum mentesség sokkal nehezebben teljesíthető, komolyabb figyelmet igényel. A továbbiakban ezt a két jelentős szempontot részletesebben vizsgáljuk

6.4.2.1 Kliens oldaliság

A program tökéletesen megfelel a kliens oldaliság követelményeinek. Működése során nem támaszkodik online erőforrásokra, miden műveletet a kliensben végez el, az internet felé kizárólag védett információt szolgáltat.

6.4.2.2 Önállóság

A program nem igényel semmilyen olyan szolgáltatást, programot, könyvtárat vagy komponenst, melyet ne tartalmazna, leszámítva a futtatásához nélkülözhetetlen Firefox böngészőt. Az össze harmadik személytől származó funkció be lett emelve a programba, telepítésekor automatikusan a vele kompatibilis komponenseket tartalmazza, a felhasználónak nem kell törődnie az esetleges függőségekkel.

A program működéséhez nem szükséges szerver oldali módosítás, így a program a szerver tudomása, engedélye nélkül is használatba vehető, tehát a program ennek a feltételnek is megfelel.

6.4.2.3 Univerzalitás

A program tesztelése során felsorolt szolgáltatásoknál nem találtunk olyan esetet, amikor az üzenet átvitele meghiúsult volna, esetlegesen kisebb kompromisszumokra kényszerültünk. Ezeket a kompromisszumokat a következő pontban részletezem.

Kijelenteni azt, hogy egy olyan program, mely ebben az internet változatos és sokféle közegében dolgozik, tökéletesen univerzális igen felelőtlen és megalapozatlan kijelentés lenne. Nagyon nagy rá az esély, hogy létezik olyan szolgáltatás, amely, például hibás struktúrája miatt, nem kompatibilis a programmal. Az program az alapvető szabványoknak megfelelő szolgáltatásokkal a teszteredmények szerint hibátlanul működik, az elsődleges cél pedig az, hogy minél szélesebb körökhöz eljusson, tehát alapvetően a népszerű szolgáltatások támogatására kell a hangsúlyt helyezni.

Ahhoz, hogy egy programot, a fenti korlátozások tudatában, univerzálisnak nevezhessünk, biztosítanunk kell, hogy az a szolgáltatások túlnyomó többségének közel összes szolgáltatásával kompatibilisek legyünk. Ilyen mértékű tesztelésre a fejlesztés során nem volt lehetőség, erre kiváló alkalmat nyújt azonban egy esetleges publikus tesztelés meghirdetése, amely mindenképpen szükséges ahhoz, hogy a program tökéletesen elássa funkcióját. A publikus tesztelés során a programot sokan, sokféleképpen, sok helyen és sok mindenre használják, így esetleges hibái, hiányosságai felszínre kerülhetnek. Ameddig a program nem esik át egy ilyen nagy volumenű tesztelésen addig nem lehet kijelenteni, hogy univerzális. Ennek ellenére elmondhatjuk, hogy a teszteredmények alapján a program várhatóan közel univerzálisnak tekinthető, a fent felállított korlátozások mellett.

6.4.2.4 Kompromisszum mentesség

A program a felhasználó számára közel kompromisszum mentesnek tekinthető, a követelményekben megfogalmazott minimális interakción felül egyetlen plusz interakcióra lehet szükség, amely a felhasználó komfortérzetét csökkenti, ez pedig az, hogy a felhasználónak egyes szolgáltatásoknál manuálisan kell előidéznie a visszafejtést.

Ezen kívül mindenképpen megemlítendő korlátozás a titkosított szöveg hosszának és az eredeti üzenet hosszának nagy eltérése, amely komoly problémát jelenthet olyan felhasználásban, amikor a beviteli mező mérete korlátozott.

A program jelenleg csak angol nyelven elérhető, hiszen így juthat el a legtöbb felhasználóhoz, de hosszú távon mindenképpen szükséges lehet lokalizált verziók kiadása. Ennek megvalósításához a keretrendszer igen jó lehetőségeket biztosít.

A felhasználó számára problémát jelenthet még, hogy a titkosított blokkok szabad szemmel jól érzékelhetőek, esetlegesen felmerülhet az igény arra, hogy az üzenet rejtett módon tárolódjon.

A jelszó menedzsment felület ezekkel ellentétben egyelőre nélkülözi a kényelmi szolgáltatásokat, kizárólag a szükséges funkciókkal rendelkezik.

Hosszútávon mindegyik probléma kezelésre szorul, ezeket a lehetőségeket a Hosszú távú fejlesztések című fejezetben tárgyalom.

6.5 Hosszú távú fejlesztések

Annak ellenére, hogy a program funkcionalitása az eltervezettekhez mérten teljesnek tekinthető még több olyan lehetőség van benne, melyeket hosszú távon érdemes lehet kiaknázni a szoftver értékének növelése érdekében. Elsősorban a program kompromisszum-mentességének növelése fontos. Ezért elsődlegesen az előzőekben említett problémák kiküszöbölésére kell koncentrálni.

6.5.1 Automatizáltság növelése

Annak elérésére, hogy a felhasználónak a lehető legritkábban kelljen a Kényszerített visszafejtéshez nyúlnia, (Optimális esetben csak szerkesztő módban) a legegyszerűbb megoldásnak az oldal késleltetett feldolgozása tűnik. Amennyiben az oldal teljesen betöltődött, már nem jelent problémát a tartalom feldolgozása. Összességében általános hosszú távú célként fogalmazható meg, hogy a felhasználónak a lehető legkevesebbszer kelljen a Kényszerített feloldást használnia.

6.5.2 Rövid üzemmód

A titkosított szöveg hosszával kapcsolatos problémák megoldására hosszú távon alkalmas lehet egy rövidített módú titkosítás bevezetése, mely valamilyen módon lerövidíti a keret hosszát, valamint nem alkalmazza a titkosított szöveg előtt a „blogcrypt” előtagot, így téve lehetővé, hogy a lehető leghosszabb hasznos teherrel ruházzuk fel a védett tartalmat. Tovább növelhető a hatékonyság, amennyiben olyan titkosítása algoritmusokat alkalmazunk, melyek rövidebb kimenetet produkálnak, vagy a bevitt szöveget esetlegesen előtömörítjük, a helymegtakarítás végett.

6.5.3 Többnyelvűség

A program többnyelvűsítése hosszú távon elkerülhetetlen, szerencsére a Firefox igen hatékony támogatást nyújt a többnyelvűsítéshez, a nyelvi fájlok elkészítése után a megfelelő

nyelv kiválasztásáról már a keretrendszer gondoskodik. A programot alapvetően elegendő néhány, elterjedtebben használt nyelvre lefordítani, ugyanis amennyiben a program elterjed, a kisebb nyelvekre a fordításokat általában már a közösség elkészíti, így a fejlesztőnek csak alkalmaznia kell azt.

6.5.4 A titkosított blokkok elrejtése

A titkosított blokkok elrejtése, mint azt már a követelményekben említettük, a kliens oldaliság megtartása mellet szteganográfiával oldható meg. A méretkorlátos beviteli mezők esetén azonban a szteganográfia alkalmazhatósága is korlátokba ütközik, valamint ügyelni kell az esetleges kódolásokból, nem várt szerver oldali transzformációkból adódó problémákra. Ilyen probléma lehet például, hogy képeken alkalmazott szteganográfia esetén az adatok kinyerése azért hiúsul meg, mert a szerver a feltöltött képet átkódolta, így elveszett a rejtett információ.

6.5.5 A jelszó adatbázis és jelszó kezelés funkcióinak bővítése

A legtöbb fejlesztést a jelszó adatbázis kezelésén lehet végrehajtani. Az adatbázis jelen pillanatban heap struktúrájú, azaz nélkülöz mindenféle rendezést, így a keresés is nehezen valósítható meg benne. A felhasználók számára a program kényelmesebb használatát tenné lehetővé, ha a jelszavak rendezetten listázódnának, lehetne őket szűrni, illetve keresni, újrasorrendezni. Ezen funkciók megvalósításához az adatbázist szükséges lenne SQLite[44] alapokra helyezni, így az automatikusan támogathatná a keresést, szűrést és rendezést, adatbázis funkcióin felül. Ekkor természetesen meg kell vizsgálni, hogy az adatok védelme és a keresés hogyan valósítható meg egyszerre és milyen feltételek mellett. Ehhez elsősorban a jelszó menedzsment modul (fileio.js) tejes cseréjére és a jelszó kezelő kiegészítésére van szükség.

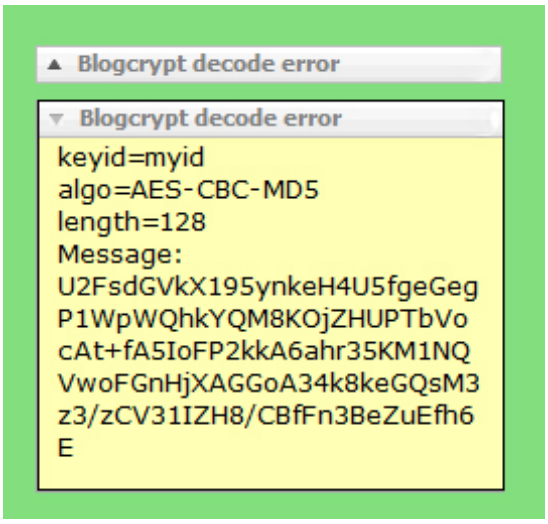
A jelszókezelés testreszabhatóságát is lehet fejleszteni. A program lehetőséget adhat arra, hogy a mesterjelszó egyáltalán ne tárolódjon, illetve arra, hogy ne kerüljön törlésre a Firefox jelszókezelőjéből.

Hasonló funkciókat lehetne megvalósítani a jelszavak kezelésénél, a jelenlegi permanens tárolás helyett szintén alkalmazható lenne időszakos tárolás, illetve a tárolást ki is lehetne kapcsolni.

[44] Az SQLite egy önálló, kisméretű (kb. 500 KiB), C forrású programkönyvtárként (library) megvalósított ACID-kompatibilis relációs adatbázis-kezelő rendszer, illetve adatbázismotor.

6.5.6 A hibák felhasználó barátabb jelzése

Szükséges lehet még a hibákat a felhasználó számára olyan módon jelezni, hogy a visszafejtetlenül maradt blokkok nem maradnak láthatóak, csak egy viszonylag szolid hibaüzenet jelenik meg, a felhasználónak pedig lehetősége van megtekinteni a paramétereket. Megoldható lenne például egy oldalba injektált szkirpttel, amelynek az alábbi tervhez hasonló eredményt kéne produkálnia.



6-5. ábra - Egy lehetőség a hibásan dekódolt mező rejtésére

A terven megfigyelhető, hogy alap esetben egy kisebb bezárt blokk látszik, benne a hibakóddal, lenyitása után, pedig az összes szükséges információ.

A project sikere esetén megfontolandó, hogy a programot más böngészők alatt is elérhetővé tegyük, így teremtve meg a lehetőséget a még széleskörűbb használatra.

7 Összegzés

Úgy gondolom, hogy a fejlesztés eredményeképp egy olyan program jött létre, mely alkalmas a kijelölt problémák megoldására és hatékony segítséget nyújthat a felhasználóknak privátszférájuk védelmét illetően, kezükbe helyezve a hozzáférés vezérlés feletti kontrollt, akár még a szolgáltatókkal szemben is.

A tervezéskor és implementáláskor felállított alapelvek betartása a későbbi fejlesztéseket könnyebbé, a kódot áttekinthetőbbé teszi, így teremtve meg az alapokat ahhoz, hogy a program hosszú távon sikereket érhessen el, a kompatibilitás megőrzése mellett viszonylag rövid fejlesztési ciklusok mellett.

Az előző pontban említett fejlesztések elvégzése hosszú távon mindenképpen elkerülhetetlen, ugyanis nagy mértékben növelik a program értékét, hiszen gazdagítják a program funkcionalitását, kényelmesebbé, egyszerűbbé teszik használatát.

Összességében tehát elmondható, hogy a program a követelményeknek megfelel, a szükséges szolgáltatásokkal rendelkezik, és a tesztelt oldalakon azokat stabilan üzemelteti is. Egy kimerítő tesztelést követően, amely például egy publikus béta teszt[45] formájában valósítható meg, a program akár arra is alkalmassá tehető, hogy egy széles körben terjesztett és használt rendszerré váljon, melynek további fejlesztésére és támogatására közösségi igény van, tehát a befektetett munka megtérül.

Ehhez azonban mindenképpen szükséges, hogy kialakuljon egy olyan stabil felhasználói bázis, akik életben tartják az igényt a fejlesztésekre és a folyamatos támogatásra. Ezt az igényt a közösségekben meg kell teremteni, hiszen a ma embere általában nem elég tudatos személyes adatait illetően és bár a megoldott probléma a jelen problémája, a megoldásra az általános igény még nem jelent meg, így a felhasználók szemszögéből egy előre mutató a jövő igényeit kielégítő alkalmazás született.

[45] Egy szoftver vagy hardver kibocsátás előtti tesztelésének utolsó fázisa, a fejlesztőkön kívüli személyek bevonásával, valós működési környezetekben.

Ábrajegyzék

Táblázatjegyzék

Irodalomjegyzék

[1]. **International Telecommunication Union.** Internet user penetration rates worldwide and for developed and developing regions, between 1997 and 2007. *International Telecommunication Union.* [Online] http://www.itu.int/ITU-D/ict/statistics/maps.html.

[2]. **O'Reilly, Tim.** What Is Web 2.0. *http://www.oreilly.com.* [Online] 2005. 9 30. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

[3]. *Anonim-e az anonim böngésző?* **Gulyás, Gábor György.** 2006. március, Alma Mater, old.: 9-30.

[4]. **Kocsis, Éva és Szabó, Katalin.** Dinamikus árazás az elektronikus piactereken. *Közgazdasági szemle.* 2002.

[5]. **Csizmazia, István.** Már krómozott az OTP adathalászok horgászbotja. *Antivírus blog.* [Online] 2009. 11 16.
http://antivirus.blog.hu/2009/11/16/mar_kromozott_az_adathalaszok_horgaszbotja.

[6]. *A web bug technológia – barát, vagy ellenség?* **Hullám, Gábor.** 2005.

[7]. **Gulyás, Gábor.** Facebook - a nagy testvér figyel téged . *Pet-portál.* [Online] 2009. 06 28. http://pet-portal.eu/blog/2009_06_28_Facebook_a_nagy_testver_figyel_teged/.

[8]. **Bucsay, Balázs.** Rycon.hu. *Rycon.hu.* [Online] 2008. 06 20. http://rycon.hu/papers/01xss.pdf.

[9]. Rycon.hu. *Rycon.hu.* [Online] 2008. 06 20. http://rycon.hu/papers/02xssworm.pdf.

[10]. *Alkalmazás-független anonimizáló rendszerek áttekintése.* **Szili, Dávid.** Alma Mater, old.: 253-280.

[11]. **Lerman, Kristina.** Social Networks and Social Information Filtering on Digg. 2006.

[12]. **Zorz, Mirko.** How social networking can hurt you. *Net Security.* [Online] http://www.net-security.org/article.php?id=1324.

[13]. **Johnson, Caroline Y.** Project 'Gaydar'. *The Boston Globe.* [Online] 2009. 09 20. http://www.boston.com/bostonglobe/ideas/articles/2009/09/20/project_gaydar_an_mit_experiment_raises_new_questions_about_online_privacy/.

[14]. **HVG.hu.** Kirúgták őket: az internet lett a vesztük . *HVG.hu.* [Online] 2009. 08 14. http://hvg.hu/Tudomany/20090813_facebook_elbocsatas.aspx.

[15]. **Hengartner, Urs, Luo, Wanying és Xie, Qi.** *FaceCloak: An Architecture for User Privacy.* 2009. augusztus.

[16]. **Guha, Saikat, Tang, Kevin és Francis, Paul.** *NOYB: privacy in online social networks.* 2008.

[17]. **Johnson, Neil F. és Jajodia, Sushil.** Exploring Steganography: Seeing the Unseen. 1998.

[18]. **Pidder.com.** pidCrypt - a Javascript crypto library. *Pidder.com.* [Online] https://www.pidder.com/pidcrypt/.

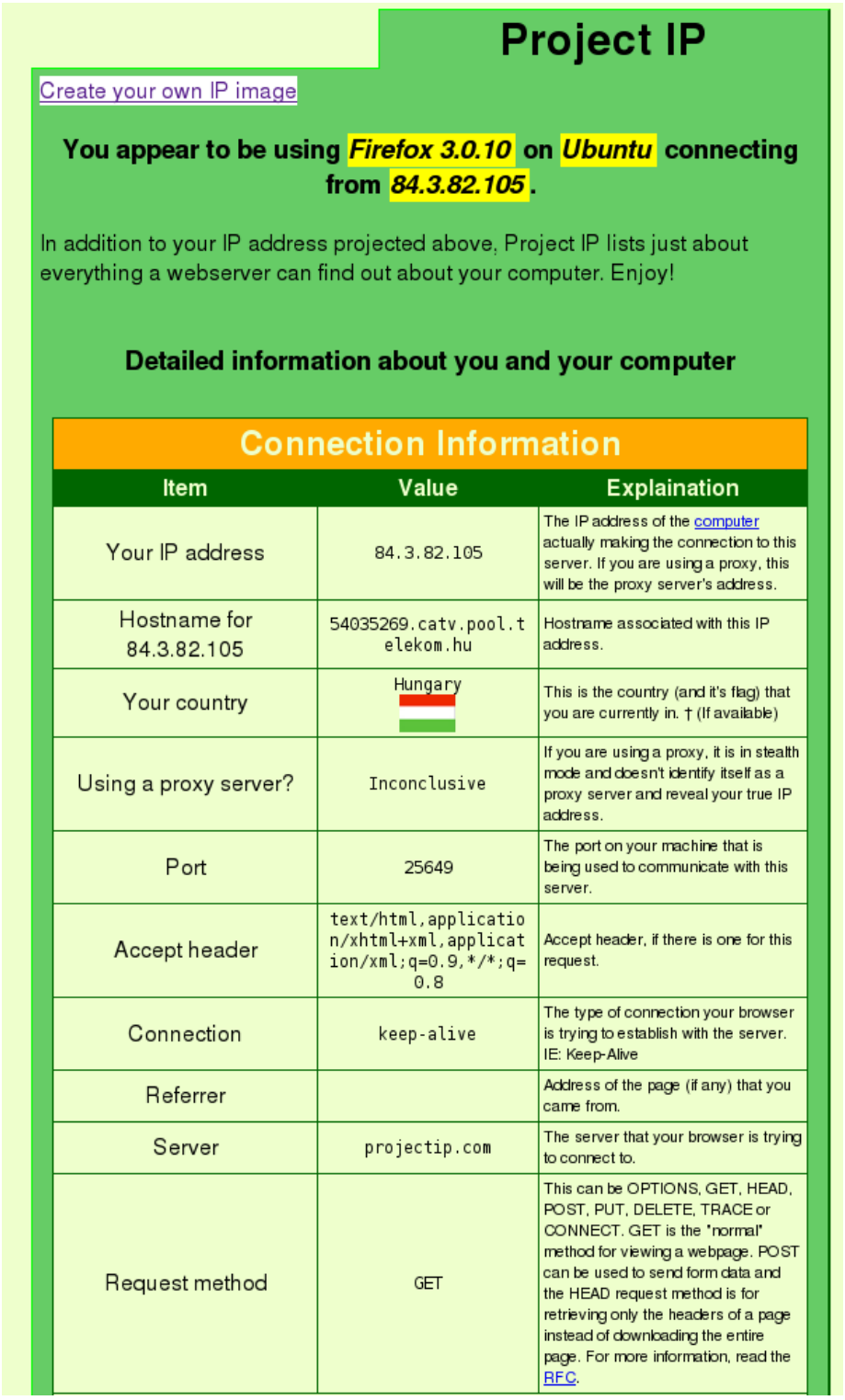
[19]. **Soltani, Ashkan, és mtsai.** Flash Cookies and Privacy. *SSRN.com.* [Online] 2009. 8 10. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1446862.

[20]. **Gulyás, Gábor, Schulcz, Róbert és Imre, Sándor.** *Comprehensive Analysis of Web Privacy and Anonymous Web Browsers.* Joint SPACE and TIME International Workshops : ismeretlen szerző, 2008.

Mellékletek

1. A projectip.com által listázott információk

Ebben a mellékletben a projectip.com által kilistázott adatokat láthatjuk. A listázott adatok valóban megfeleltek a tesztkörnyezet adatainak.



Project IP

Create your own IP image

You appear to be using *Firefox 3.0.10* on *Ubuntu* connecting from *84.3.82.105*.

In addition to your IP address projected above, Project IP lists just about everything a webserver can find out about your computer. Enjoy!

Detailed information about you and your computer

Connection Information

Item	Value	Explanation
Your IP address	84.3.82.105	The IP address of the computer actually making the connection to this server. If you are using a proxy, this will be the proxy server's address.
Hostname for 84.3.82.105	54035269.catv.pool.telekom.hu	Hostname associated with this IP address.
Your country	Hungary	This is the country (and it's flag) that you are currently in. † (If available)
Using a proxy server?	Inconclusive	If you are using a proxy, it is in stealth mode and doesn't identify itself as a proxy server and reveal your true IP address.
Port	25649	The port on your machine that is being used to communicate with this server.
Accept header	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8	Accept header, if there is one for this request.
Connection	keep-alive	The type of connection your browser is trying to establish with the server. IE: Keep-Alive
Referrer		Address of the page (if any) that you came from.
Server	projectip.com	The server that your browser is trying to connect to.
Request method	GET	This can be OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE or CONNECT. GET is the "normal" method for viewing a webpage. POST can be used to send form data and the HEAD request method is for retrieving only the headers of a page instead of downloading the entire page. For more information, read the RFC.

About Your Browser

Item	Value	Explanation
Browser	Firefox	This is what your browser identified itself as. This may be incorrect because many (especially newer) browsers have options to change how the browser identifies itself, allowing the browser to masquerade as a different one.
Browser version	3.0.10	The version of your browser- what more can be said?
Gecko based?	Yes	Is your browser a Gecko-based browser? Gecko is an HTML-rendering engine from Mozilla that is used in a wide variety of browsers such as: Firefox, Camino and Netscape
Gecko version	Feb 26, 2016 (2009042523)	The release date of the version of the Gecko rendering engine used by your browser.
Mozilla compatibility level	5.0	Most browsers state their compatibility level as a Mozilla version.
Character set	ISO-8859-2,utf-8;q=0.7,*;q=0.7	The client's preffered charset. For example: iso-8859-1
Encoding	gzip,deflate	Types of encoding (compression) that your browser can accept. For example: gzip This can make a big difference in download time and bandwidth usage.
Encrytion strength	128 bit	The US Government had a ban on exporting encryption stronger than 40-bit, so major browsers had different versions of encryption for release in the US and other countries. The US has since relaxed the ban, and most browsers ship with high encryption to all countries.
Language	hu-hu,hu;q=0.8,en-us;q=0.5,en;q=0.3	This is the preferred language of the browser. It usually consists of two parts, for example "en-au", indicating English as the language, and Australian English as the secondary language. Some websites look at this header and change the language of the page according to the accept language. In many browsers you can change the HTTP Accept Language if you look deep enough in the options.
User agent string	Mozilla/5.0 (X11; U; Linux i686; hu-HU; rv:1.9.0.10) Gecko/2009042523 Ubuntu/8.10 (intrepid) Firefox/3.0.10	This identifies the make and model of your browser. Some websites use this information to indirectly know the capabilities of your browser and change (or tailor) the webpage for your best browsing experience. Some browsers actually allow you to modify the User Agent. For example the Opera browser Allows you to change its identity to Microsoft Internet Explorer or Mozilla, and if you're using Firefox, there's a great extention, the User Agent Switcher, available to modify the UA.

Platform (OS) Information

Item	Value	Explaination
Operating system	Linux	The platform or operating system (OS) running on your computer.
Operating system version	Ubuntu	The version of your OS, if available.

Scripting and Plugins

Item	Value	Explaination
Javascript enabled	Yes	Is the client-side programming language "JavaScript" enabled in your browser?
JavaScript version	1.8	Your browser supports this version of JavaScript.
Java enabled	No	Is the Java browser plugin installed and enabled?
All Plugins	• Default Plugin • Demo Print Plugin for unix/linux • Totem Web Browser Plugin 2.24.3 • Windows Media Player Plug-in 10 (compatible; Totem) • DivX® Web Player • QuickTime Plug-in 7.2.0	What plugins do you have installed in your browser? Does not work in Internet Explorer on Windows due to a deficient implementation of Javascript.

Screen Properties

Item	Value	Explaination
Window size	610 x 301 pixels	The width and height of your browser window's client area. *
Maximum window size	1,280 x 677 pixels	Window size when your browser is maximized. *
Screen size	1,280 x 677 pixels	The width and height of your computer screen. *
Color depth	24 bits	1 bit=> Black and white 2 bits=> 4 colors 4 bits=> 16 colors 8 bits=> 256 colors 16 bits=> 32,768 or 65,536 colors 32 bits=> 16,777,216 colors *
Colors	16,777,216	How many colors your dispaly/OS can handle. * * The accuracy of the screen property items is dependant on your browser's implementation of JavaScript, which often leaves much to be desired.

Miscellaneous Info

Item	Value	Explanation
Cookies enabled?	Yes	Is your browser configured to accept cookies?
Anti-aliasing fonts	No	Is font smoothing currently enabled.
Pages viewed	6	How many pages have you visited during this browser session? If you are using a tabbed browser (for example: Firefox, Opera, Netscape 6, or MSIE 7) this is how many pages you have viewed **in the current tab**.
Your timezone	GMT+0100 (CET) You observe Daylight Saving Time	Your machine's timezone settings, as reported by Javascript.
Clipboard contents (text)		The last text item you copied onto your clipboard! Only works in Internet Explorer 6 on the Windows platform. It reportedly works with varied success when IE is running in an emulator such as VMWare on another OS. If you have to use Windows, at least dump IE and use Firefox. Rogue, evil websites can use this to steal potentially sensitive data from your Windows clipboard. I have done this in Javascript within the browser and the contents of your clipboard is not sent to this server. If someone wanted to snoop they would do what I have done, except the text area where it's displayed would be invisible (using CSS `display: none;`) and they would use an XMLHttpRequest object to send it back to the webserver, all without your knowledge. **Fix:** Go to *Tools > Internet Options > Security > Select a security zone > Custom Level > Scripting > Allow paste operations via script* and set it to *Disabled* or *Prompt*.

2. A tervezés során meghatározott funkciók és a hozzájuk rendelt interfészek

Az alábbiakban azokat a specifikációs részleteket ismertetem, melyek a konkrét tervezés során igen fontos szerepet játszottak, a feladat és az implementáció megértése szempontjából azonban nem annyira jelentősek, hogy a dolgozat törzsében helyet kaphattak volna.

Ezek a táblázatok a tervezési fázisban leírtakkal megegyező sorrendben tárgyalják az egyes modulok funkció-igényeit és a hozzájuk rendelt interfészeket, valamint az egyes kiegészítő táblázatokat.

Mező	Funkció
minVersion	A legkisebb támogatott FireFox verzió – Jelen esetben 3.*
maxVersion	A legmagasabb támogatott Firefox verzió – Jelen esetben 3.5.*
id	A kiegészítés egyedi azonosítója, konvencionálisan „kiegészitésneve@fejlesztőneve.domain” formátumú, ahol a domain a készítő nemzetiségének megfelelő domain utótag. Jelen kiegészítés esetében például „blogcrypt@paulikt.hu”.
name	A kiegészítés neve – A konkrét implementációban: Blogcrypt
version	A kiegészítés verziószáma: 1.0
creator	A fejlesztő neve: Tamas Paulik
description	A kiegészítés leírása: „Experimental project”
homepageURL	A kiegészítés hivatalos honlapja: http://pet-portal.eu/?page=blogcrypt_intl"
contributor	A kiegészítés fejlesztésében segédkező személyek felsorolására: Gabor Gulyas
optionsURL	A kiegészítés beállításait tartalmazó XUL állomány elérési útja. (chrome://blogcrypt/content/options.xul)

Az install.rdf fontosabb mezői

Fő Funkciók	Függvény
Eseménykezelők beállítása (a fájl betöltődésekor)	addEventListener(eventType, listener, useCapture)
Inicializálás (init esemény)	init()
Megszűnéskori műveletek (uninit esemény)	uninit()
Automatikus feloldás indítása, ha nincs szerkesztő mód (onload esemény)	onLoad(aEvent)
Szerkesztő mód váltása (A gomb oncommand attribútumán keresztül kerül meghívásra)	switchEditorMode()
Kézzel indított feloldás indítása (A gomb oncommand attribútumán keresztül kerül meghívásra)	passmanager()
Segédfunkciók	**Függvény**
A beállítások inicializálása	initperfs()

A blogcrypt.js funkciói

Fő Funkciók	Függvény
Visszafejtés a {titkos szöveg, jelszó, titkosítási metódus} hármasból	decrypt(crypted, password, method)
Titkosítás a {nyílt szöveg, jelszó, titkosítási metódus} hármasból	encrypt(plain, password, method)
Segédfunkciók	**Függvény**
Az AES-CBC-MD5 mód inicializálása és visszafejtés a {titkos szöveg, jelszó} páros alapján	aes_cbc_md5_decrypt(crypted, password)
Az AES-CBC-MD5 mód inicializálása és titkosítás a {nyílt szöveg, jelszó} páros alapján	aes_cbc_md5_encrypt(plain, password)
Az AES-CTR mód inicializálása és visszafejtés a {titkos szöveg, jelszó} páros alapján	aes_ctr_decrypt(crypted, password)
Az AES-CTR mód inicializálása és titkosítás a {nyílt szöveg, jelszó} páros alapján	aes_ctr_encrypt(plain, password)
Pidcrypt titkosítási funkciók	Pidcrypt függvények

Az encryption.js funkciói

Fő Funkciók	Függvény
Jelszó felvétele az adatbázisba	add_passwd_to_db(domain, keyID, password)
Jelszó kikérése az adatbázisból	get_passwd_from_db(domain, keyID)
A teljes jelszó-adatbázis elkérése, táblázat szerű mátrix reprezentációban, ahol egy sor egy jelszó, a sor elemei pedig a {domain,keyid,jelszó} mezők	get_passwd_db()
Jelszó törlése az adatbázisból	delete_passwd_from_db(domain, keyID)
Segédfunkciók	**Függvény**
A jelszó adatbázist reprezentáló string kiolvasása és visszafejtése	read_key_db()
A jelszó adatbázis reprezentáló string titkosítása és eltárolása	write_key_db(dbstring)
A mesterjelszó elkérése a Firefox jelszókezelőjétől. Amennyiben a jelszó még nem volt eltárolva, akkor annak bekérése és eltárolása	get_master_passwd()
A mesterjelszó törlése a Firefox jelszókezelőjéből	remove_master_passwd()

A fileio.js funkciói

Fő Funkciók	Függvény
A titkosítandó blokk kinyerése az oldalból, a BBCodeok feloldásának indítása, majd a titkosítási paraméter-választó megnyitása	enCrypt(indocument)
A titkosítás befejezése. Ezt a metódust hívja a bezáródó jelszó választó. Ha a szöveg egy szövegdobozból származik, akkor lecseréli azt, ha máshonnan származik, vágólapra teszi.	enCrypt_second_part (selection, method, keyID, domain,node)
Segédfunkciók	**Függvény**
A BBCodeok feldolgozása	parseBBCode(selection) – külső forrásból származik a bbcode.js tartalmazza

Az encryption.js függvényei

Funkció	XUL elem(ek)
Minta a kijelölt szövegből	<description>
Jelszavak megjelenítésének ki-/bekapcsolása	<checkbox>
Jelszó választó	<radiogroup> és <radio>
Algoritmus választó	<radiogroup> és <radio>

A pswdchooser.xul elemei és funkciójuk

Fő Funkciók	Függvény
A betöltés lekezelése, a grafikus felület dinamikus feltöltése. (Futását az ablak betöltődése idézi elő)	init()
A jelszó megjelenítés állítását végző függvény. (Futását a checkbox megváltozása idézi elő)	showPasswords()
Az OK gombra kattintást lekezelő függvény. Ha minden paraméter ki van töltve, akkor elindítja a titkosítás második szakaszát, ha nincs, akkor nincs funkciója. (Futását az OK gombra kattintás idézi elő)	doOK()
A Cancel gomb megnyomását kezelő függvény. Mindenféle hatás nélkül bezárja az ablakot. (Futását az OK gombra kattintás idézi elő)	doCancel()
Segédfunkciók	**Függvény**
Hozzáad egy jelszót egy felsoroláshoz	addkeytoRadioGroup(keyid,password,radioname)
Kiüríti a jelszavak felsorolását	clearRadioGroup(radioname)
A kulcsadatbázis alapján feltölti a jelszavak listáját.	parsekeydb()

A pswdchooser.js funkciói

Fő Funkció	Függvény
A fentiekben leírt funkcionalitást valósítja meg	deCrypt(doc)

A decrypt.js funkciói

Funkció	Megvalósító XUL elem(ek)
Tájékoztató felirat: („Keys stored in the database”)	<caption>
A jelszavak listája	<listbox>, <listhead>, <listcols>, <listcol>
„Delete”, „Import”, „Export All”, „Export selected”, „Add” funkciók	<button>
Új jelszó beviteli mező feliratai	<label>
Új jelszó beviteli felület	<textbox>
A jelszavak megjelenítése	<checkbox>

A passwordmanager.xul elemei és funkciójuk

Fő Funkciók	Függvény
A felület inicializálása és a jelszólista feltöltése.	init()
Új jelszó felvitele a kitöltött mezők alapján. A kitöltött mezők érvényességének ellenőrzése a hozzáadás előtt.	addkeytodb()
Kiválasztott jelszó törlése az adatbázisból.	deletekeyfromdb()
A kiválasztott elemek exportja, az exportálási jelszó bekérése, az exportálási adatok elhelyezése a vágólapon.	passdbexport()
A teljes adatbázis exportja, az exportálási jelszó bekérése, az exportálási adatok vágólapra helyezése	passdbexportall()
Az importálási jelszó bekérése és az adatok importálása a megadott exportálási adatok alapján	passdbimport()
Segédfunkciók	**Függvény**
Jelszó hozzáadása a listához	addkeytolist(domain,keyid,password)
A jelszó lista kiürítése	clearListbox()
A jelszólista beolvasása és az elemek elhelyezése a listában, egyesével, az addkeytolist funkciót használva	parsekeydb()

A passwordmanager.js funkciói

3. Fejlesztőkörnyezet választása Javascripthez

A fejlesztés megkönnyítése végett olyan eszközt kerestem, amely képes az automatikus kiegészítésre, a futási hibák kiírására és rendelkezik szintaxis kiemeléssel. Több fejlesztőkörnyezet kipróbálása után végső választásom a Notepad++[46] szövegszerkesztőre esett. Automatikus kiegészítéssel ugyan nem rendelkezik, de megfelelő és testre szabható szintaxis kiemelési funkciókat biztosít.

A hibaüzenetek kezelésére a Console[2] [47] Firefox kiegészítést alkalmaztam, mely a hibaüzeneti konzolhoz enged hozzáférést, keresési, szűrési funkciókkal támogatva.

A fejlesztés során használtam még a QuickRestart[48] kiegészítést, amely a Firefox egyszerű és gyors újraindítására ad lehetőséget. Az újraindítási funkció azért fontos, mert a források módosítása után a változások a Firefoxban csak a kliens újraindítása után jelennek meg.

A weboldalak struktúrájának elemzésére a DOM Inspector 49 kiegészítést használtam. A weboldalak tartalmának feldolgozásakor keletkező hibák analízisénél nyújtott segítséget, mert képes voltam megtalálni azokat az elemeket a honlapban, melyek a problémákat okozták

Hasznos eszköznek bizonyult ezen felül az Extension Developer[50] kiegészítés, mely több hasznos eszközt is a fejlesztő rendelkezésére bocsájt. Ezek közül a Regexp Evaluator volt számomra a leghasznosabb, meg a program fejlesztése során szükséges reguláris kifejezések ellenőrzését könnyíti meg.

4. Fejlesztőkörnyezet választása XULhoz

A XUL felületek fejlesztéséhez is a Notepad++-t használtam. Ehhez is rendelkezésre állt szintaxis kiemelés, de ez a funkció egyes esetekben, ismeretlen okokból nem működött. A XUL kódok hibáinak felderítésére az Extension Developer XUL Editorját használtam.

---

[46] http://notepad-plus.sourceforge.net/hu/site.htm

[47] https://addons.mozilla.org/hu/firefox/addon/1815

[48] https://addons.mozilla.org/hu/firefox/addon/3559

[49] https://addons.mozilla.org/hu/firefox/addon/6622

[50] https://addons.mozilla.org/hu/firefox/addon/7434