1 Popis

Sdružení CZ.NIC, správce národní domény, se zabývá zabezpečováním provozu domény nejvyšší úrovně .CZ.

Každý den se mnoho stránek stává obětí hackerských útoků. Ty mimo jiné zanechávají v kódu persistentní škodlivý obsah, který představuje riziko pro návštěvníky kompromitovaných stránek - snaží se zneužít slabiny v jejich prohlížečích a neaktualizovaných doplňcích, přimět uživatele k nainstalování dalšího malware nebo vylákat přístupová hesla k službám třetích stran.

CZ.NIC, kde autor textu pracuje, se snaží hackerské aktivitě zamezit, aktivně vyhledává napadené stránky a kontaktuje jejich majitele s upozorněním, že k nákaze došlo a dalšími informacemi, jak její důsledky řešit. K vyhledávání těchto prezentací těží data z veřejně dostupných zdrojů škodlivých domén a o každém incidentu si vede záznamy.

1.1 Motivace

Abychom přinášeli majitelům relevantní informace, napadená stránka je podrobena analýze, která má za cíl zjistit, zda je nákaza skutečně přítomná, nebo se jedná falešný poplach.

Tato analýza se nezřídka dělá vícekrát, např. když majitel domény žádá informaci, zda je už nákaza vyřešená. Protože škodlivý kód, který napadá stránky, může mít mnoho forem, je jeho lokalizace časově náročná.

Lze sice užívat veřejně dostupné skenery webových stránek. Praxe ukazuje, že pro množství napadených stránek, které je třeba analyzovat, nejsou tyto skenery zcela spolehlivé. V mnoha případech bylo třeba další manuální práce, opatrné, aby si administrátor sám návštěvou zavirovaných stránek nenakazil počítač.

Je žádoucí správu napadených webových stránek co nejvíc automatizovat.

Cílem této práce je nejprve poskytnout přehled o škodlivém kódu, který který zanechávají útočníci v kódu napadených webových prezentací. Navrhnout a naprogramovat aplikační řešení, které je schopné na základě vlastní analýzy vyhledávat tento škodlivý kód, zjistit, co tento kód dělá a na jaké IP návštěvníka unáší. Toto řešení má být připojeno k stávajícímu, které CZ.NIC používá, a šetřit lidský čas, spojený se správou napadených stránek. Aplikaci uvedeme do provozu. Abychom ověřili, že je řešení v pořádku, provedeme testování v reálném provozu.

1.2 Organizace práce

Nejprve popíšeme, jak práce s incidenty v současnosti probíhá a ukážeme průchod systémem, který se používá. Potom důkladně rozbereme práci administrátora tohoto systému a zmíníme slabiny, které práci zpomaluje. V analytické části probereme druhy škodlivého kódu a teoreticky i na příkladech ukážeme, co tento kód dělá a jak se na stránky / k návštěvníkovi mohl dostat.

Navrhneme pak vhodné řešení, které tento kód bude schopno samo na stránkách vyhledávat a provedeme jeho implementaci. Abychom zjistili, jak se systém osvědčuje, otestujeme průchod systémem s uživatelem i bez uživatele, které bude mít za cíl zjistit, zda program je v praxi schopen škodlivý kód najít.

V diskuzi probereme zajímavé poznatky, se kterými jsme se v práci setkali a zmíníme možnosti dalšího rozvoje.

1.3 Terminologické poznámky

Protože hlavním jazykem programátora je v současné době angličtina, budeme v práci používat několik termínů. ke kterým v českém jazyce neexistuje ekvivalent, nebo by srozumitelnost textu jejich otrokyckým přeložením utrpěla. Rigorózní čtenář je pak prošen, aby jejich užití v českém jazyce omluvil.

- cache, cachovat, default, exploit, framework, log, loggovat, mailing list, parser, sandbox, screenshot, ticket, tracker, URL

2 Pozadí problematiky

Ke správě informací o napadených doménách využívá CZ.NIC aplikaci Malicious Domain Manager (MDM), která byla vyvinuta v roce 2012 výzkumným pracovištěm ve spolupráci s národním týmem pro řešení bezpečnostních incidentů (CSIRT[1]), který je CZ.NIC provozován. MDM autonomně prohledává veřejně dostupné zdroje informací o doménách napadených některým typem škodlivého kódu a informuje o nich administrátora[CZ.NIC, 2012]. Nabízí rozhraní, skrze které lze zahájit komunikaci s majitelem domény, a mailové vlákno ukládá do trackovacího systému Request Tracker.

V následující kapitole si předvedeme funkčnost MDM, obeznámíme čtenáře s úlohou administrátora a zjistíme, jaké má slabiny stávající proces správy napadených domén.

2.1 Stávající řešení

MDM z hlediska aplikační architektury tvoří dva oddělené systémy, oba napsané v Pythonu2 a sdílející přístup k PostgreSQL databázi. První z nich obstarává vnitřní funkcionalitu - prohledává dostupné internetové zdroje, hlídá v nich změny a obohacuje databázi. Druhý z nich je přístupný přes webový server a slouží pro vlastní administrátorskou práci v aplikaci.

Zdrojové kódy jsou veřejně dostupné[2] pod svobodnou licencí GNU GPL v3.

2.1.1 Vnitřní funkcionalita

Plánovač úloh na hostujícím serveru periodicky spouští dva skripty, které představují bránu k vnitřní funkcionalitě aplikace. Ta je znázorněna na obrázku.

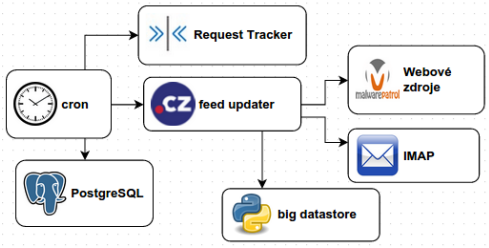


Figure 1: Diagram vnitřní funkcionality

První ze skriptů obsluhuje spojení se serverem, na kterém běží Request tracker. Souhrnně zanáší změny do databáze. Jeho hlavním účelem je, aby zrychlil načítání webového serveru, který může tickety sledovat z kopie v lokální databázi, místo aby se připojoval k Request trackeru s každým webovým požadavkem. Je doporučeno jej spouštět každých deset minut.

Druhý skript se volá jednou za půl hodiny. Iniciuje dotaz na dostupné zdroje, zda neobsahují hlášení o nově napadených doménách. K webovým zdrojům se připojí přes http protokol a přes mailový protokol IMAP se prohledává mailing list. Aby nezatěžoval zdroje tím, že by opakovaně načítal celou jejich databázi, výsledky vyhledání se cachují

[1] http://csirt.cz
[2] https://gitlab.labs.nic.cz/labs/mdm

v lokálním úložišti (na obrázku big datastore), obsluhovaným Pythonem. V současné době MDM využívá Malware Patrol, Google Safebrowsing, Phishtank a Zeus Tracker.

2.1.2 Webový server

Webové rozhraní MDM běží na serveru Apache jako Python modul, napsaný ve frameworku Pyramid[3].

Na hlavní stránce se nabízí souhrnné statistické údaje o počtu incidentů (nových / řešených / vyřešených) a seznam nově došlých e-mailů, které zaslali majitelé domén. V dolní části je seznam e-mailů, u nichž se nepodařilo dohledat mailové vlákno, kam patří, protože jejichž odesílatelé změnili předmět. Na následujícím obrázku vidíme screenshot hlavní strany,.

Vlevo na každé stránce je menu, které umožňuje procházení různých kategorií incidentů a vyhledávací pole.

Podívejme se například do kategorie Nové hrozby. V této kategorii jsou incidenty, které ještě administrátor neřešil. V prvním sloupci nalezneme seznam nakažených domén. Ve sloupci Současné hrozby je uveden počet nakažených URL, evidovaných k doméně. Dále tabulka obsahuje informace o předpokládaném typu malware, čas, kdy byla hrozba naposledy zjištěna a zdroj, ze kterého informace pochází. Pro lepší orientaci umožňuje tabulka stránkování a změnu počtu zobrazení výsledků na stránku.

Kliknutím na název domény se dostaneme na stránku s detailním popisem incidentu. V tabulce jsou vypsána všechna nakažená URL, ostatní sloupce se podobají výpisu kategorie. Na kartě historie administrátor vidí seznam URL, která už byla webovým zdrojem vyhodnocena jako bezpečná, na kartě korespondence je pak formulář, skrze který může napsat vlastníkovi. Formulář obsahuje několik šablon mailu. Mail obsahuje formální žádost o odstranění nákazy, seznam napadených URL a podrobnosti, které se v šabloně liší podle typu zdroje.

Karta status obsahuje interní záznam o incidentu, kdy byl vytvořen, vyřešen a jaký administrátor s ním pracoval. Interní záznam se generuje v prostém textu, do kterého může administrátor zasáhnout a napsat vlastní poznámku. Jak vidíme na obrázku, některé weby jsou napadány znovu a znovu.

Figure 2: Úvodní stránka webového rozhraní

[3]http://docs.pylonsproject.org/projects/pyramid/en/latest/

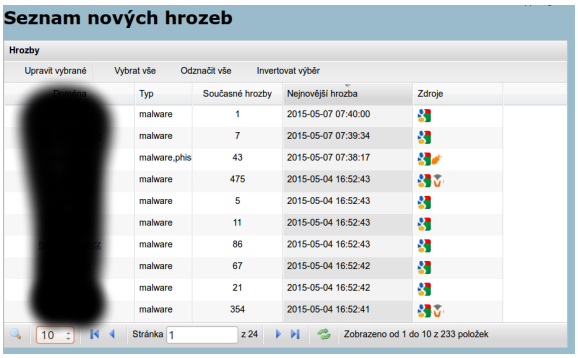


Figure 3: Výpis kategorie incidentů

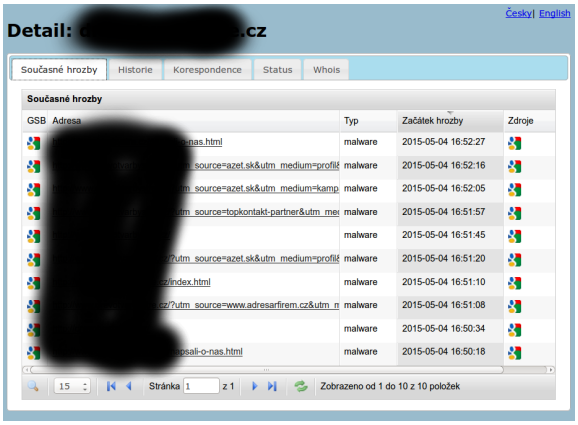


Figure 4: Detail incidentu domény - karta současné hrozby

2.2 Úloha administrátora

Nyní probereme činnost, kterou vykonává administrátor. Jsou to 2 typy úkonů - kontakt s majitelem nakažené domény a další analýza napadené stránky.

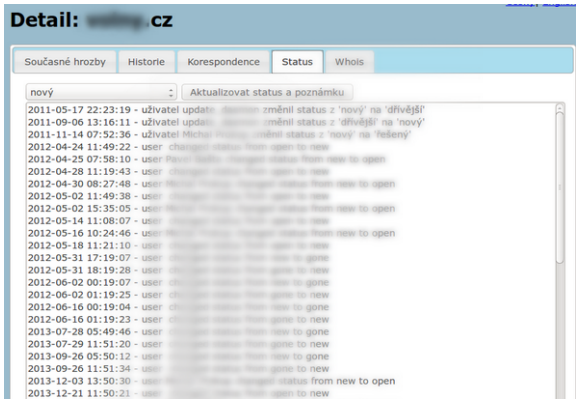


Figure 5: Karta status incidentu

2.2.1 Komunikace s vlastníkem

Administrátor informuje vlastníka domény o zjištěném útoku, aby ten mohl co nejrychleji zasáhnout, přijmul opatření, která zabrání zneužití uniklých dat a minimalizoval vzniklou škodu. V kontaktním modulu nástroje MDM administrátor vybere šablonu podle původu hlášení nákazy - např. šablona pro databázi Phishtank obsahuje link k ticketu, který se o jeho doméně v databázi vede, pro Google Safebrowsing šablona obsahuje návod co dělat, aby vyhledávač přestal doménu blokovat. Kontaktní údaje vyhledá v interním registru domén, které CZ.NIC spravuje, nebo přímo z napadených stránek. V momentě, kdy administrátor odešle první mail, se incident přesouvá do kategorie Řešené, kde zůstává, dokud není webová prezentace vyhodnocena jako znovu bezpečná.

Často se stává, že se mail vrátí jako nedoručitelný. Tehdy se administrátor snaží najít jiný kontaktní údaj správce domény.

Figure 6: Hlavní činnosti administrátora

2.2.2 Analýza stránky

Hledají se stopy, odkud útočník přišel, resp. s kterými škodlivými adresami je útok spjat. Data se pak v současnosti využívají hlavně ve službě Turris [footnote], výzkumném projektu, který pomáhá uživatelům s ochranou domácí sítě pomocí speciálního routeru.

2.3 Slabiny v procesu

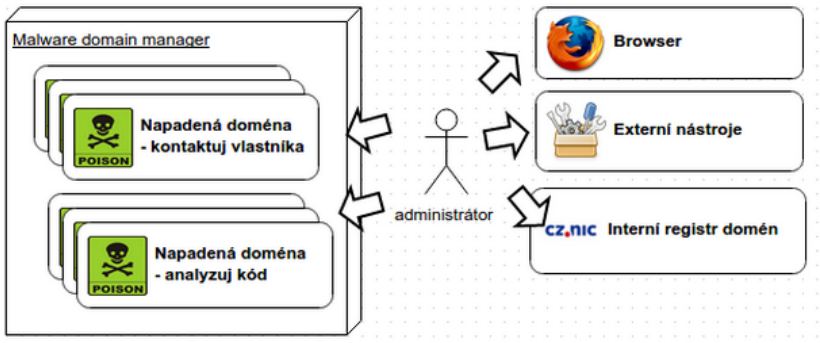
Počet incidentů se rok od roku zvyšuje, od doby spuštění služby MDM zaznamenalo už přes 8 000 nakažených domén. Každý den jich 5 - 30 přibývá.

Hlavní problémy procesu tkví (1) ve vysokých nárocích na organizační schopnosti administrátora a (2) na velkém počtu akcí, které je administrátor nucen dokola provádět.

Protože je třeba ruční analýzy napadeného webu, ať už v lokálním prohlížeči (zabezpečeným pluginem) nebo ve virtuálním operačním systému, administrátor neustále čelí zvýšenému riziku, aby se jeho pracovní stanice sama nestala obětí ještě neznámého exploitu, který bude na jedné ze stovek stránek nasazen. Část analýzy lze přenechat pomoci automatických nástrojů (např. UrlQuery, Sucuri SiteCheck), ty ovšem při větším počtu analýz náhodně selhávají. Analýzu stránky přitom nelze vynechat, naopak je třeba ověřit, že podezření na přítomnost malware není mylné, a někdy je třeba analyzovat stránku vícekrát za sebou, abychom se ujistili, zda nákaza už je pryč.

Opakované akce Úloha administrátora kromě vlastní analýzy napadeného webu spočívá v tom, že údaje o každé doméně proklikne, najde si kontaktní e-mail a odešle vlastníkovi šablonu. Zvláště dohledat si kontaktní e-mail v interním registru domén znamená navštívit až sedm po sobě jdoucích stránek. Pakliže vlastník domény nereaguje a situaci nenapraví, je na administrátorovi, aby hlídal, zda měl vlastník dost času na odpověď a zda existuje jiný komunikační kontakt, jiný mail či telefon; a aby zjistil, nemá-li zákazník plný mailový koš. Pole s textovými poznámkami k doméně v kartě Status, které nahrazuje CRM, je zastrčené ve vlastním tabu a jeho správa a udržování vyžaduje další klikání a skrolování. Pokud doména byla v historii napadena vícekrát za sebou a tedy několikrát měnila stav z *vyřešeno* na *nová hrozba a* zpět, situace je zvlášť nepřehledná. S přispěním toho, že prostředí MDM nativně neumožňuje třídit domény do vláken podle více parametrů (zda už proběhla jejich analýza a zda vlastník byl úspěšně kontaktován), v praxi pak vznikají zmatky, vedoucí k tomu, že se tytéž stránky analyzovaly opakovaně, nebo se jednomu kojícímu majiteli volá třikrát za sebou.

Administrátor má tedy k dispozici pokročilý nástroj, ale stále na něm zůstává břemeno dalších manuálních činností (viz obrázek) - vyhledávání kontaktu vlastníka a časově náročná analýza stránky v prohlížeči či skrze externí nástroje.



3 Analýza

Probrali jsme, jakým způsobem v současnosti probíhá řešení bezpečnostních incidentů a kde jsou slabá místa. Abychom mohli navrhnout způsob, jak proces urychlit, podíváme se hlouběji na to, co konkrétně bezpečnostní incident obnáší - v čem nákaza škodlivým kódem spočívá, jak se projevuje a jaká známá řešení by nám mohla být nápomocna.

3.1 Druhy škodlivého kódu

Rozlišujeme mnoho kritérií, podle kterých můžeme škodlivý kód klasifikovat - například dle způsobu nákazy, účelu či vedlejších účinků.

Některý škodlivý kód se posílá poštou, jiný se replikuje do souborů na disku. Adware je dobře patrný podle toho, že zobrazuje reklamy, ransomware zločinně šifruje data na disku, keylogger odposlouchává hesla a příkazy a trojan působí, že počítač oběti se stane zombie pěšákem[Lis,].

Viry mohou být psány v žertu, první známý virus na světě jenom zobrazoval na počítači zprávu "Chyť si mě", virus Cruncher dokonce působil kladně - komprimoval soubory a šetřil tak místo. [kaspersky]

Dnes už ale hrozbu malware nikdo nepodceňuje. Denně vznikají desetitisíce nových algoritmů. Jenom v roce 2013 vzniklo 20 % malwaru[Pan, 2014]. Malware používají hackeři i vlády, aby nelegální cestou získávali osobní, finanční a firemní informace. Dochází dokonce k případům vydírání - hackeři požadovali výkupné po slovenském poskytovateli Vnet, aby ustali se svým DDoS útokem, který blokuje dostupnost služeb. Vnet se musel vypořádat s úctyhodným tokem napadených serverů 400 Gb / s, což je "niekoľkonásobne väčšie množstvo dát, než preteká celým slovenským internetom"[Slo, 2015].

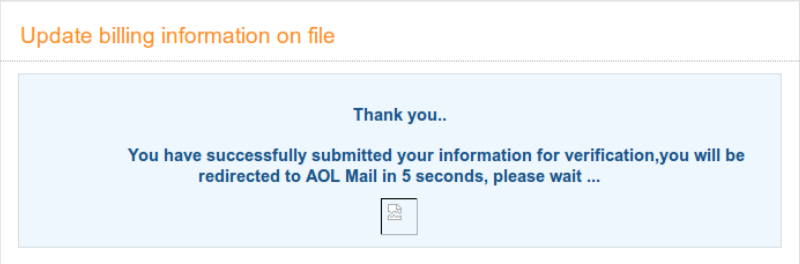
Pro náš účel nás zajímá pouze škodlivý kód, který ohrožuje návštěvníky webu a útočí buďto na prohlížeč (malware), nebo na návštěvníka (phishing). Takové útoky se nezaměřují jen na známé redakční systémy a frameworky, kde jedna objevená zranitelnost ohrožuje tisíce implementací naráz, ale i na lokální aplikace či statické HTML stránky. Útoční boti zjistí, jaké služby na serveru běží, zkouší prolomit hesla FTP serveru, případně přístup dostanou přímo od webmastera, jehož počítač byl napaden virem. Útoky na webové stránky přicházejí ve vlnách. Stránky vlastněné týmž člověkem bývají napadnuty v rozmezí několika minut po sobě, byť spolu obsahově ani technologickou strukturou nesouvisejí.

3.1.1 Phishing

Cílem phishingu je vylákat od návštěvníka citlivé údaje, často přihlašovací heslo do nějaké služby. Využívá přitom důmyslné maskovací techniky. Na napadeném serveru vytvoří stránku, která věrně imituje přihlašovací formulář. Zbytek serveru může phishingový útočník nechat zcela netknutý a plně funkční, takže si webmaster při návštěvě

svých stránek nemusí všimnout ničeho podezřelého. Nebo oběť přesměruje na stránku, jejíž URL připomíná adresu legitimní napodobované webové prezentace.

Na obrázku níže vidíme důvěryhodně vypadající zprávu, která se nachází na jednom českém serveru. Náhodný návštěvník je přesměrován na adresu, která připomíná přihlašovací formulář služby AOL.



Pokud návštěvník vlastní AOL účet a přihlásí se, jdou údaje přímo útočníkovi. Ba co víc, útočníkův skript může s nabytými údaji uživatele skutečně přihlásit na jeho účet, takže si uživatel nedozví, že jeho heslo bylo ukradeno.

Jak ukazuje výzkum[Mohebzada et al., 2012], velké procento uživatelů je náchylné útoku podlehnout. Výzkumníci legální cestou získali e-maily studentů univerzity, zaregistrovali si stránku *http://www.myaus.info*, která se podobá oficiálním stránkám *http://myaus.aus.edu* a rozeslali 10 000 phishingových e-mailů. O heslo přišlo 9 % studentů a 5 % zaměstnanců univerzity.

Přesto se phishing pozná velmi snadno. Pokud máme echo, že se na stránce vyskytuje, poznáme jeho přítomnost většinou na první pohled.

3.1.2 Malware

Forma škodlivého kódu se v nahlášených doménách opakuje, protože se v drtivé většině nejedná o osobní průlom hackera, ale o stopu po některém z automatických nástrojů. V javascriptové šabloně nebo konkrétní stránce na webu se inkluduje kód, často velmi obfuskovaný (nečitelný člověkem), který vposledku na pozadí vyvolává *http* požadavek. Některý kód čeká na určitou verzi operačního systému nebo prohlížeče[Kolbitsch et al., 2012]. Pozná ji podle http hlaviček nebo dle veřejně známých hacků, čili detailů, v kterých se chování prohlížečů liší. Jiný malware se objeví pouze jednou, zkusí provést škodlivou činnost a do prohlížeče uloží *cookie*. Pokud se návštěvník vrátí v témž prohlížeči s návštěvní cookie, malware se na stránce vůbec neprojeví a není možná jeho detekce.

Způsob napadení návštěvníka Nástrojem útoku na webových stránkách jsou především skriptovací funkce prohlížeče. Útočník vloží do stránky javascriptový kód, který je schopen vykonávat libovolnou škodlivou činnost na pozadí. Pokud ho nevloží přímo na stránky, zneužije některý HTML tag, který ho donačte.

Zde uvádíme přehled často zneužívaných/zneužitelných cest, které malware využívá, podle zdroje [Phakoontod and Limthanmaphon, 2012, 67-8]. Přitom u laskavého čtenáře předpokládáme znalost jazyka HTML a javascriptu.

HTML tagy:

- tag *script* - buď přímo obsahuje kód, který prohlížeč vykoná, nebo pomocí atributu *src* ukazuje na URL se škodlivým kódem
- iframe, frame - vloží do stránky jinou stránku
- embed, object - vloží do stránky objekt, například škodlivý Flash
- img - načte obrázek z jiné domény (proč je to škodlivé, vysvětlíme níže)
- meta - prostě unese návštěvníka na jinou doménu

Do libovolného tagu můžeme vložit kód v atributu - tzv. *inline javascript*. Tagu *b*, který dělá text tučným, nastavíme atribut *onclick* (nebo *on-cokoli*), který při kliknutí na element spustí javascript událost.

> text

Toto se zvláštním způsobem se týká navíc nevinného tagu *link*, určeného pro načítání stylů, kde můžeme javascript vložit do atributu *href*.

Zneužívané funkce javascriptu:

- document.location - přesměruje návštěvníka pryč
- document.cookie - vloží škodlivou url
- document.write - přepíše stránku
- document.createElement - vytvoří dynamicky HTML element
- alert - testovací funkce, jejíž přítomnost na produkčním webu signalizuje, že byl zneužit
- unescape - překládá kódovaný řetězec do písmen
- eval - spustí libovolný kód z textového řetězce

Maskování kódu O funkci eval se zdroj[Phakoontod and Limthanmaphon, 2012]kupodivu nezmiňuje, přesto tuto mocnou funkci, v kombinaci s funkcí *unescape*, škodlivý kód nezřídka vyžívá. Umožňuje proměnit nevinně vypadající textový řetězec v dynamicky generovanou funkci. Ta pak není vidět ve zdrojovém kódu stránky, protože v době jejího generování funkce ještě neexistovala. Nezkušený webmaster proto ve svém zdrojovém kódu vidí jen nic neříkající řadu znaků a nerozpozná, že se jedná o škodlivý kód. Pokud rozpozná, není snadné zjistit, co kód přesně dělá bez toho, aniž ho spustí, což je riskantní.

Ukrytá funkce může vypadat například takto:

```
> <script>eval('\x66\x75\x6e\x63\x74\x69\x6f\x6e\x20\x77 ...
```

Abychom kód bezpečně přeložili do původních příkazů, můžeme funkci eval nahradit funkcí console.log a spustit v konzoli prohlížeče.

```
> console.log('\x66\x75\x6e\x63\x74\x69\x6f\x6e\x20\x77 ...
```

Zjistíme, že škodlivý kód vypadá takto:

```
> function wcaD(pnt){function nuEeg(bfNzv){var dBb=0;var bDKB=bfNzv.length;var oA
```

Jak vidíme, škodlivý kód využil některý obfuskátor, který přejmenoval všechny proměnné a názvy funkcí na nesmyslné řady znaků, takže tento kód snadno prohlížeč vykoná, ale člověk mu porozumí jen stěží.

Setkali jsme se s tím, že místo nesmyslných znaků obfuskátor použil slovník angličtiny a všechny objekty skriptu přejmenoval na sémanticky nosná anglická slova, takže kód na první pohled vzbuzoval dojem, že ho přímo psal člověk. Až posléze bylo jasné, že slova nemají žádný význam a pouze matou.

Jiný škodlivý kód může být obfuskován několikanásobně. Při použití console.log jsme pak nedostali prohlížečem zpracovatelný kód, ale pouze další funkci eval naplněnou řadou nesmyslných symbolů. Abychom se dostali na dřeň takovéhoto kódu, museli jsme jsme nový eval opět nahradit funkcí console.log . K našemu velkému překvapení byla funkce tímto a ještě jedním podobným způsobem v sobě zakódovaná 72×, což jsme zjistili, až když jsme opatrně vytvořili rozkódovací algoritmus na míru.

Útočníci také mohou zneužít známá jména, například jeden typ nákazy na stránku importuje soubor jquery-1.40.15.js, jehož jméno evokuje oblíbený framework. Navíc tento framework se většinou používá také v obfuskované podobě, ovšem ne z důvodu matení, ale minifikace - jsou odstraněny všechny mezery a názvy proměnných zkráceny. Neznalý člověk proto tento způsob maskování může přehlédnout.

Poslední způsob skrývání javascriptu, který uvedeme, spočívá v použití funkce setTimeout. Kód odloží svoje vlastní spuštění a unikne tak pozornosti [Rozzle de-cloaking Fig 17].

HTML elementy se nejčastěji skrývají pomocí CSS stylování, příkazy display: none, visibility:hidden, nebo nastavením šířky a výšky na nulu. Oblíbená technika útočníků je umístit tento neviditelný element pod kurzor myši návštěvníka, který na napadené nebo pirátské stránce kliká například na tlačítko spuštění videa, ovšem místo toho klikne na skrytý element (tzv. clickjacking, únos kliknutí myši).

Někdy škodlivý kód nebývá skryt vůbec. Kupodivu častým znakem malware, na který často narážíme, je šestiznakový komentář před anebo po vpašovaném kódu. Kód může vypadat například takto:

```
> document.write('<script type="text/javascript" src="http://xavieanssystems.com/
> /*/60c35e*/
```

Motivaci pro vkládání zbytečného komentáře vedle škodlivého kódu se nám nepodařilo rozluštit. Zjistili jsme však, že na tuto jasnou stopu útoku přišli i autoři antivirových systémů. Dle serveru virustotal.com, který shromažďuje analýzy dostupných antivirů,

byl předchozí příklad označen jako škodlivý kód 18/52 antivirů. Odstranili jsme potom funkci document.write a nechali jen komentář (to jest již zcela neškodný kód), avšak i tehdy byl skript nadále hlášen jako heuristický trojan.

Kromě javascriptu a HTML elementů lze maskovat i celé názvy domén. Málokterý uživatel totiž ví, že doménové jméno jde vyjádřit různými způsoby. Jak uvádí etický hacker Robert Kümmel[Kümmel, 2011, 118], na Google doménu se skrz DNS servery můžeme dostat pěti způsoby. K povšimnutí jsou především poslední tři, méně známé.

- http://www.google.cz - přímé doménové jméno
- http://74.125.87.104 - IP adresa
- http://1249728360 - dword tvar
- http://0x4a.0x7d.0x57.0x68 - hex tvar
- http://0112.0175.0127.0150 - oktalový tvar

(Počet tvarů není konečný, příklad platí pro IPv4, adresy typu IPv6 můžeme zapsat ještě v dalších tvarech.)

Další vynikající metoda, na kterou jsme v posledních měsících začali narážet, je používání mnohonásobných řádů u domény. K doméně www.fotokorri.wz.cz si například hacker nastavil čínskou doménu 7. řádu www.fotokorri.wz.cz.619f4903b9d47923.d99q.cn . Neznalý uživatel, který neví, jak se URL adresa tvoří, ji snadno přehlédne. Doména 3. řádu sestává z řady nesmyslných čísel a pokud přehlédne tečku, snadno si bude myslet, že čísla jsou jen nějaký sémanticky nevypovídající parametr.

Tuto techniku využívá veškerý typ malwaru a dle zkušenosti směřuje doména 1. řádu do Číny (cn), nebo na Kajmanské ostrovy (cc). Doména je pořízena zadarmo a může být dle potřeby útočníkem rychle zrušena, takže po jejím původním obsahu nezůstává žádná nalezitelná stopa.

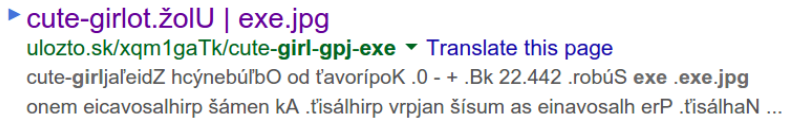
Cíl škodlivé činnosti Činnost škodlivého kódu na stránkách je v roce 2015 stejná jako v době před třemi lety, jak popisuje zdroj [safeguard]. Škodlivý kód na stránkách funguje především jako brána pro další formy nákazy.

Nejčastěji se provádí útok Drive by Download, kdy je návštěvníkův prohlížeč unesen na nepřátelskou stránku, která vyhodnotí jeho vlastnosti a podstrčí mu adekvátní exploit [browser jsguard], který se pokud možno na pozadí nainstaluje do prohlížeče oběti či přímo do jejího systému.

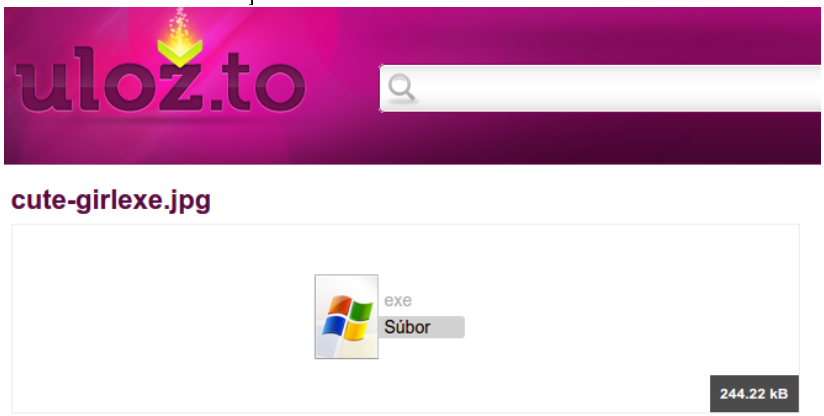
Jindy může útočník nabídne ke stažení spustitelný soubor .exe s líbivým jménem v naději, že si ho návštěvník stáhne. Velmi vynalézavou zranitelností, kterou obsahovaly starší verzi prohlížečů, je Unitrix. Útočník využil vlastnosti jednoho znaku Unicode, který se ve východních zemích využívá k přepnutí textu na čtení zprava doleva. Ke stáhnutí vystavil soubor se jménem např. Girl_Al-gpj.exe, ovšem místo pomlčky za 'AL' je umístěn tento neviditelný řidicí znak, který řetězec "gpj.exe" otočí na "exe.jpg". Návštěvník si tedy myslí, že otevírá obrázek se jménem "Girl_Al**exe.jpg**". V době

psaní práce je stále možnost na tuto zranitelnost narazit v praxi. Na soubor s takto upraveným jménem jsme narazili například na službě *Uložto.sk.*

Na obrázku níže vidíme, jak vypadá výsledek vyhledávání tohoto souboru na Google. Všimněte si, že Unicode znak přeházel metadata stránky.



[příklad maskování domény - zranitelnost Unitrix]



[cute-girlexe.jpg je ve skutečnosti .exe soubor]

V dnešních prohlížečích už byla chyba Unitrix záplatována a při stahování souboru prohlížeč ukazuje správnou příponu.

Podstrčit soubor lze však i jinými způsoby. Setkáváme se s tím, že útočník napodobí stránky Youtube, předstírá, že spustí video nějaké žhavé ženy, ale hned na to vyzve uživatele, aby "updatoval svůj Flash plugin pro přehrání videa", přičemž věrně napodobí design Adobe Flashe. Místo pluginu však návštěvník autorizuje ve svém počítači jasný vir.

Kromě stahování nebezpečných souborů hackeři útočí i jinak. Zmíníme ještě útok zjednodušeně CSRF, který využívá toho, že je uživatel v prohlížeči přihlášený v jiné službě, například ve fiktivní Bance.cz. Útočník zjistil, že Banka nepoužívá token pro autorizování požadavků platby, a tak do napadené stránky super-videa-sexy.cz umístí obrázek:

```
> <img src='http://banka.cz/ucet/poslat?komu=hacker&kolik=1000' />
```

V ideálním případě stačí, že oběť je v Bance přihlášená na vedlejším tabu prohlížeče a na tuto stránku přijde jen na okamžik, a hacker se obohatí.

Způsob napadení serveru Abychom měli obrázek o malwaru na webech ucelený, je třeba ještě probrat, kudy se nákaza dostane na server. Existuje obrovského množství známých děr a jejich počet se rozrůstá. Jen v minulém roce 2014 se objevily dvě

obrovské zranitelnosti, které zahýbaly celým internetem, protože zasahovaly i řadu důvěryhodných serverů. První z nich byl **Heartbleed** bug, chyba v šifrování SSL. Podle skenu několika desítek tisíc českých adres, který CZ.NIC provedl, se ukazuje, že na tuto zranitelnost bylo náchylných kolem 6 - 9 % serverů [Ďurechová and Duračinská, 2014]. Druhý bug **Shellshock** se týká zneužití nezamýšlené vlastnosti samotného unixového *bashe*, která v jeho zdrojovém kódu zůstávala dlouhá léta nepovšimnuta. Vzhledem k tomu, že většina serverů běží pod Unixem, šlo se prolomit do serveru pouhou správně upravenou *http* hlavičkou. Testy CZ.NIC ukazují[Ďurechová,], že jakmile byla zranitelnost zveřejněna, hackeři zahájili mohutnou vlnu útoků, které cílili na dosud neupdatované servery.

Normální *http* hlavičku *User-Agent*, kde prohlížeč posílá informace o své verzi:

```
> User-Agent: Mozila Firefox 36
```

Stačí ji upravit následovně a na náchylném serveru spustíme perl s privilegii webového serveru, který vykoná kód z naší škodlivé adresy:

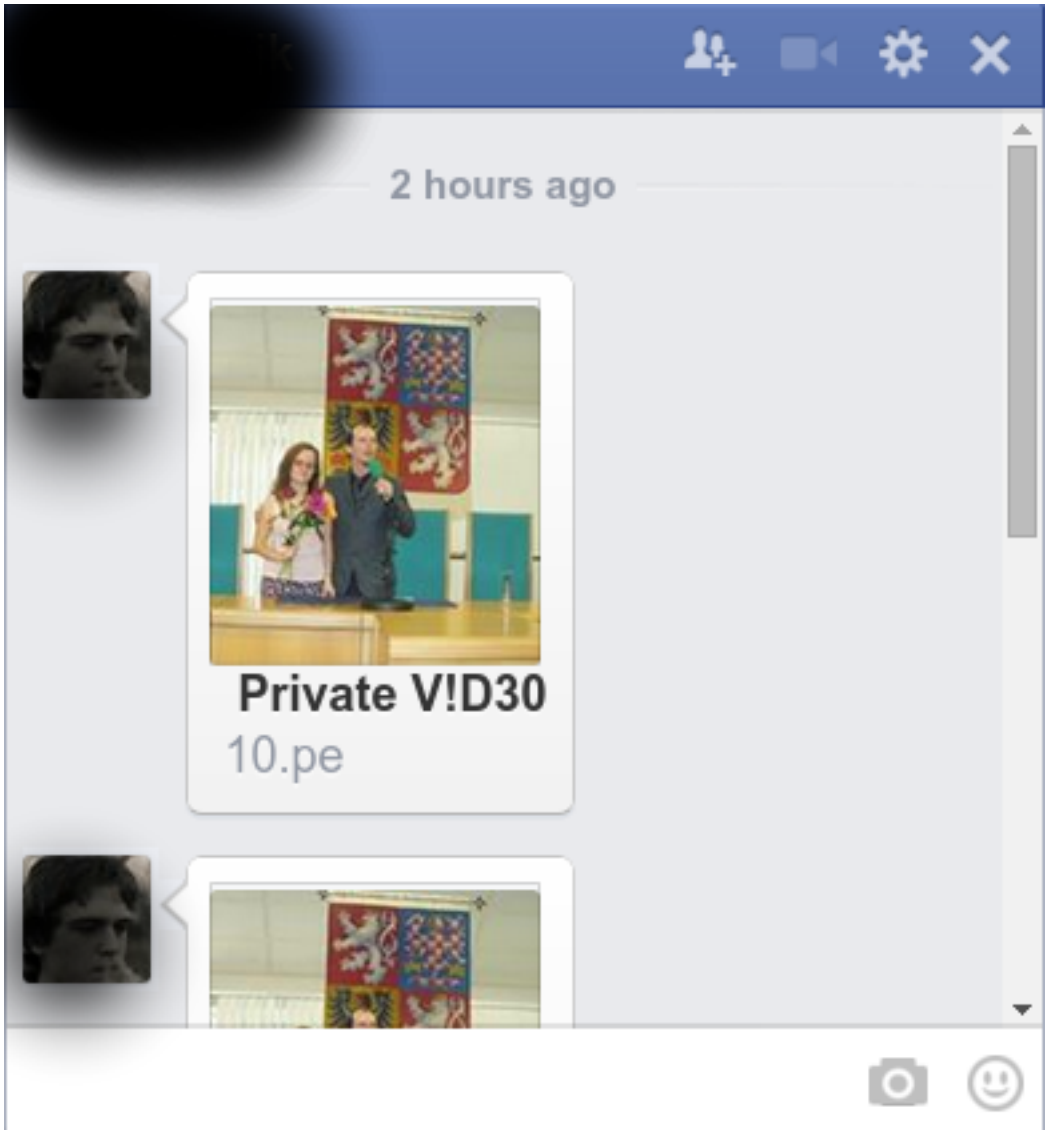
```
> User-Agent: () { :; }; curl http://202.143.160.141/lib21/index.cgi | perl
```

Stranou chyb v serveru existuje také škála chyb ve webové prezentaci. Náchylné jsou webové prezentace s framework jako je CMS Wordpress, zvlášť pokud obsahují řadu neaktualizovaných pluginů. Hackerům snadno prohledají velký adresový prostor, najdou domény, na kterých běží Wordpress a testují jeden exploit za druhým.

Klasickým způsobem útoku jsou pak neošetřené uživatelské vstupy - injekce SQL kódu a injekce javascriptu (tzv. XSS) [surfGuard].

Ovšem i bezchybně záplatovaná aplikace může být napadnuta, pokud je napadnut systém webmastera, kde má uložena přístupová hesla.
Mobily a tablety jsou v dnešní době o řád zranitelnější než notebooky a stolní počítače, jak ukazuje pokus CZ.NIC. Stačí v restauraci začít poslouchat, jaké wifi sítě zařízení vyhledávají a podle toho určit, kde se hosté ve svém soukromém životě zdržují[Bašta, 2014]. Pokud útočník podvrhne SSID některé z vyhledávaných sítí, přístroje hostů se mohou začít připojovat přes něj - a být kompromitovány.

Mezi typické patří také útoky přes sociální inženýrství. Ve Facebook chatu přichází odkaz, který se tváří jako video a má profilovou fotku cílového uživatele, aby zvýšil zvědavost oběti. Všimněme si, že odkaz nemá v popisu řetězec “video”, ale “v!d30”, aby co nejdéle mátl automatické protispammové filtry - captcha stroje proti strojům. V případě, že na odkaz klikneme, skončíme na nebezpečné stránce.



[zneužitý účet přítele na FB rozesílá scam]

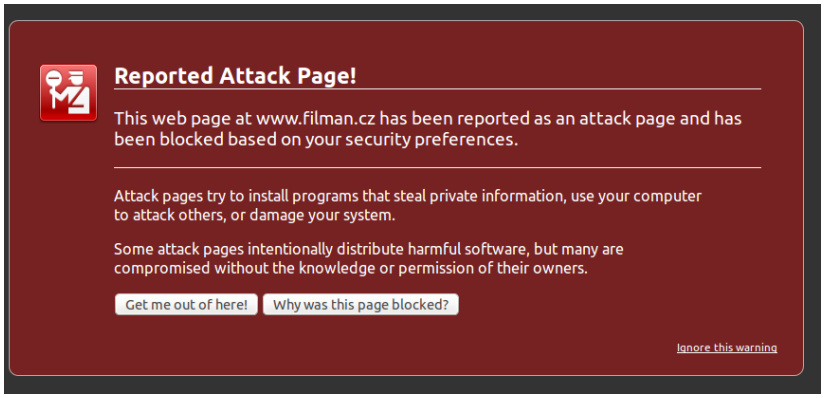
3.2 Související nástroje

Současná řešení boje proti malware operují na několika úrovních. (Využíváme poznatky ze zdroje[Sachin and Chiplunkar, 2012, 268-9].)

3.2.1 Automatické nástroje

První úroveň je antivir a firewall na počítači klienta.

Za další úroveň můžeme považovat nadějný projekt Google Safebrowsing. Pokud vyhledávač Google vyhodnotí stránku jako vadnou, zanese ji do veřejné databáze Safebrowsing. Z ní čerpají data prohlížeče Firefox a Chrome. Pokud uživatel těchto prohlížečů přijde na podezřelou stránku, zobrazí se mu varovný text.



[výstražný text google safebrowsing]

3.2.2 Volitelné doplňky v prohlížeči

Další úroveň zabezpečení na webu mohou přinést bezpečnostní doplňky. Zde zmíníme NoScript[4], který v základním nastavení blokuje veškeré nebezpečné skriptování. Uživatel každý web, který navštěvuje, explicitně označí jako povolený. Má tak jistotu, že pokud se náhodou dostane na neznámou nebezpečnou stránku, skripty se nevykonají. Podobnou funkcionalitu nabízí i doplněk JS Guard[Kishore et al., 2014].

3.2.3 Více profilů

Pokročilou úroveň zabezpečení přináší také používání více profilů v prohlížeči. Jeden profil můžeme používat na prohlížečí sociálních sítí a druhý na internetové bankovnictví. Pokud se nakazí jeden profil, významně to sníží pravděpodobnost kompromitace i druhého profilu.

3.2.4 Prevence serveru

Webmaster serveru by neměl opomenout následující kroky (dle seznamu[Scambray and Shema, 2002, 312].)

- odstranit testovací uživatele (a uživatele guest)
- správně nakonfigurovat síť (firewall serveru)
- správně nakonfigurovat a záplatovat webserver
- nastavit oprávnění v databázi (běžný uživatel nesmí mít root práva)
- ošetřit veškerý uživatelský vstup v aplikaci

[4]https://noscript.net/

Můžeme zkusit i hacknout/zkontrolovat vlastní server některým z dostupných nástrojů a služeb: [footlinks!] Wapiti (xss, sql injekce) , Havij (sql injekce), Qualys (SSL certifikát), VirusTotal, Quttera, Sucuri, urlQuery (detekce škodlivého kódu). Za zmínku stojí také projekt *shodan.io*, který vyhledává na internetu nezabezpečené *věci* - od stavu ledničky přes ovládání kluziště po veřejně dostupné kontrolní panely elektráren.

4 Návrh

Na základě předchozích znalostí se pokusíme navrhnout řešení, které urychlí stávající správu škodlivých domén. Hlavní problémy tkvěly ve zdlouhavé práci s aplikací MDM a v časové náročnosti ručního analyzování napadené stránky. Detailněji si tedy probereme, jak je možno práci s aplikací a ruční analýzu urychlit a automatizovat.

4.1 Práce s MDM

Práce s aplikací MDM by měla být rychlejší. Je třeba identifikovat sekvence kroků, které administrátor provádí vždy za sebou, a pomoci mu je přeskakovat. K hlavním problémům patří především uklikanost aplikace. Nepřináší žádné zvláštní klávesové zkratky a použití některých pluginů jQuery (například *tabs* pro dynamický výpis tabulek) znesnadňuje ovládání stránek pomocí klávesnice.

Administrátor v aplikaci provádí tři typy úkonů, a to podle aktuálních časových možností:

Posílat maily vlastníkům nově napadených domén Činnost je třeba udělat co nejdříve od obdržení hlášení, alespoň jednou denně, aby vlastníci domén mohli včas reagovat.

Naše řešení by mělo hlavně administrátorovi pomoci s hledáním kontaktů na vlastníky domén, protože to je třeba dělat každý den, i v časovém presu. Přitom samotné vyhledání v interním doménovém registru je záležitostí rutinní, však zdlouhavou, protože k nalezení e-mailu je kvůli každému kontaktu třeba znovu projít několik stránek.

Analyzovat napadené domény Tuto akci může administrátor vykonávat jednou za pár dní, může projít hromadně několik domén.

Naše řešení by měla automaticky spojovat nahlášené url s nějakým spolehlivým analyzátorem. Můžeme zkusit využít nějaký nástroj z konce minulé kapitoly (např. urlQuery nebo Sucuri), nebo vyvinout vlastní analyzátor. Analýza by měla zjistit, které škodlivé stránky s napadenou stránkou souvisí, jejich IP a případně jak konkrétní instance škodlivého kódu vypadá.

Telefonicky kontaktovat vlastníky, kteří nejsou dostupni na mailu Majitelé domén mají v doménovém registru často neaktuální informace a maily zaslané na oficiální kontaktní adresy se vrací zpátky. CZ.NIC v takovém případě nemůže promptně poskytnout informaci, že je web v nepořádku. Administrátor by se však měl snažit kontaktovat vlastníky i jinak, nejrychleji telefonem, pokud má majitel v registru nastaven telefonní údaj (telefonní čísla dle praxe zastarávají pomaleji než mailové adresy).

Tato činnost se musí vykonávat přes den, nelze v noci.

Naše řešení by mělo z interního registru domén načíst telefonní údaj.

Shrnutí Protože 1. a 3. činnost souvisejí (obojí načítá údaje z interního registru), zkusíme vyvinout software, který spojí MDM s interním registrem. Platí však omezení, že ke zdrojovému kódu interního registru nemáme přístup, je zabezpečen zaměstnaneckým jménem a heslem, není tedy možné vytvořit API, které by s MDM komunikovalo. Protože je však současně počet uživatelů MDM omezen na administrátory z řad zaměstnanců společnosti, můžeme vyvinout standardizovaný doplněk, který interní registr přiměje s MDM komunikovat.

Zkusíme navrhnout, jak by vypadal tento doplněk, a pak budeme řešit 2. činnost, propojení MDM s analyzátorem.

4.2 Doplněk do MDM a interního registru

Sekvenci úkonů, která spočívá v mačkání stejných tlačítek na známých stránkách a doplňování hodnot do vstupních polí, může výborně udělat uživatelský javascript. Doplněk Greasemonkey pro Firefox umožňuje spouštět uživatelské javascripty na různých stránkách. Řešením by bylo, kdyby tento doplněk na požádání rozklikal hlášení o všech incidentech, zjistil, jestli už proběhla mailová komunikace, a pokud ne, zahájil by ji se šablonou dle typu incidentu a na adresu automaticky získanou z interního registru.

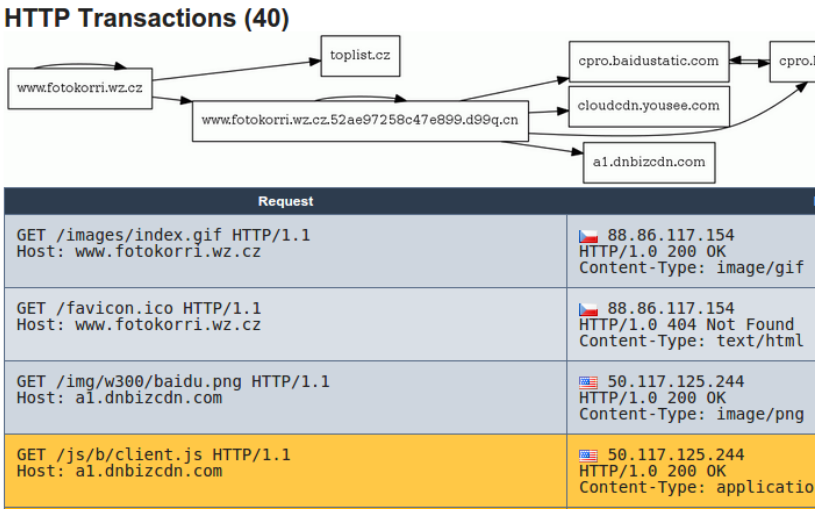
Doplněk musí vyřešit komunikaci mezi zabezpečeným interním registrem a MDM.

4.3 Automatizace analýzy

Jako doprovodný nástroj k ruční analýze administrátoři využívají systémy urlQuery a Sucuri. Oba systémy proskenují url a vrátí výpis škodlivého kódu, který na stránce naleznou. Ukážeme si, jak vypadá jejich výstup. Budeme testovat na výsledcích analýzy nakažené stránky *www.fotokorri.wz.cz*, o které z ručního testování víme, že obsahuje přesměrování na stránku *www.fotokorri.wz.cz.8e07b6066bc6b549.d99q.cn* .

4.3.1 Nástroj urlQuery

Analýza ve službě urlQuery trvá asi 45 vteřin. Výpis obsahuje screenshot, výpis škodlivých příkazů spouštěných javascriptem a seznam domén, na které se stránka připojila. V diagramu vidíme, že čínskou doménu urlQuery korektně našlo.

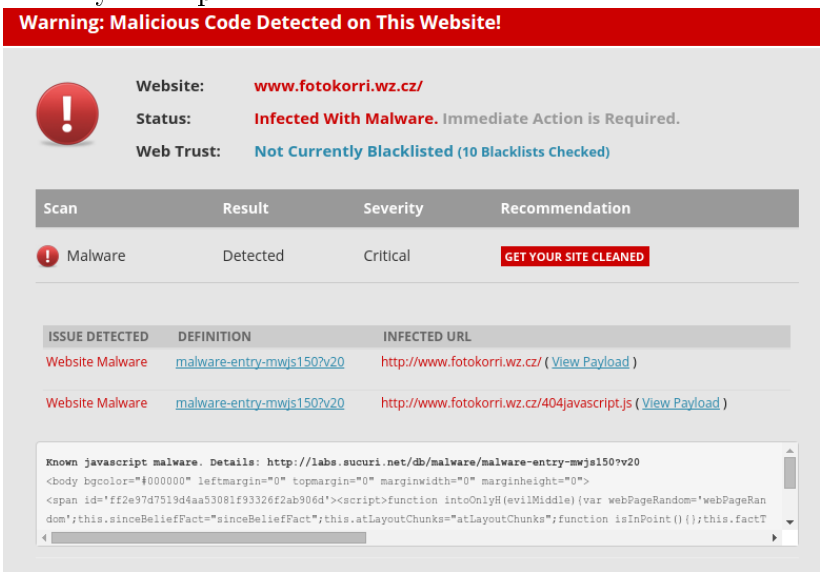


[výpis urlQuery]

4.3.2 Nástroj Sucuri

Analýza v Sucuri je o něco rychlejší, ale má o něco horší výsledky a neobsahuje screenshot. Jak vidíme v náhledu výsledků, korektně ukazuje, že je stránka kompromitovaná, ale méně často najde domény. V našem případě ve výsledcích nezobrazuje čínskou doménu, ovšem ukazuje blok javascriptu, kde se škodlivý kód nachází (ovšem méně přehledně než ve službě urlQuery, která přímo zobrazuje, jaké škodlivé příkazy se provedou).

Služba Sucuri je sympatická tím, že přináší webmasterům webu tipy, co dělat, i pokud je jejich skenovaná webová prezentace v pořádku, například upozorňuje na neaktualizovaný Wordpress.

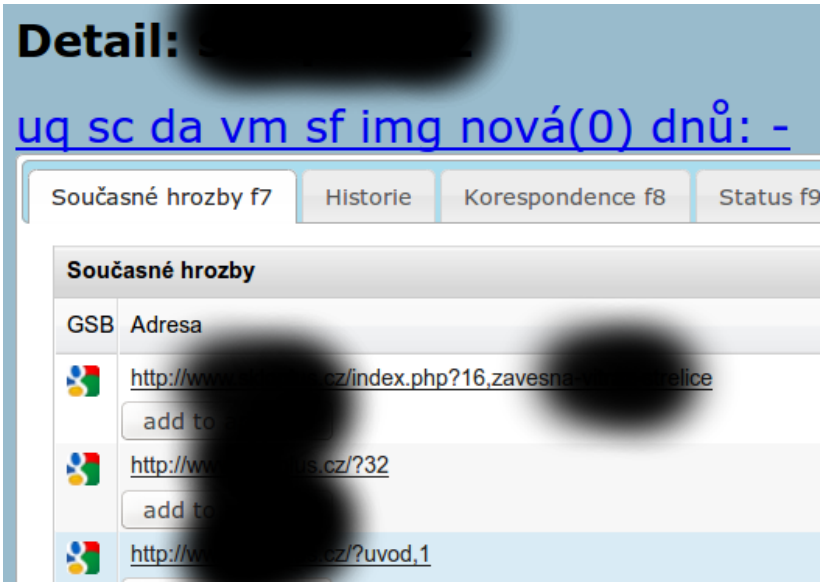


[výpis sucir]

4.4 Integrace nástrojů

Pokusili jsme se oba nástroje spojit se službou MDM. Do stránky s výpisem detailu incidentu MDM jsme vložili automaticky generované linky na zmíněné analyzátory, které se při otevření stránky samy spouštěly ve vedlejších tabech prohlížeče. Lokální prohlížeč byl pomocí uživatelských skriptů upraven tak, aby analyzátory vzaly URL z MDM a bez meškání spustily generování skenu, aniž by člověk musel mačkat tlačítko "ok".

(Na následujícím screenshotu vidíme automaticky generované linky, všimněme si "uq" pro urlQuery a "sc" pro Sucuri. Další linky, o kterých bude řeč níže, jsou "vm" pro virtuální stroj, "sf" pro Google Safebrowsing, "da" pro interní registr domén, "img" pro službu zajišťující screenshot a "nová" pro automatické odesílání zpráv majiteli.)



[předběžná implementace automatických nástrojů do MDM]

Experimenty ukázaly, že urlQuery vrací vynikající výsledky, ovšem analýza trvá příliš dlouho a při větším množství analýz beznadějně padá, analyzuje se donekonečna. Během práce s MDM by bylo vhodné moci analyzovat třeba 10 URL najednou, ovšem urlQuery toho nebylo schopno. Z neznámých důvodů urlQuery selhalo třeba i při jediné analýze a místo výsledku vrátilo oznámení, že stránka není dostupná. (Důvody nelze s jistotou určit, protože neznámé vnitřní strukturu urlQuery, roli může hrát i geografická vzdálenost od analyzovaného serveru.)

4.4.1 Virtuální stroj

Přestože jsme spustili dva externí analyzátory, administrátor musel nakonec přeci jen vyhodnotit většinu stránek ručně.

Aby přitom zůstal v bezpečí, měl by mít vypnuté skriptování v prohlížeči alespoň na úrovni doplňku NoScript. Jenže když se škodlivý kód nevykoná, jeho ruční analýza je natolik zdlouhavá, že se stává nepoužitelnou.

Testovali jsme proto možnost spouštět URL ve virtuálním stroji, skrz aplikaci Virtualbox [footlink]. V něm běžel starý operační systém WinXP a ničím nechráněný Firefox. Malware se sice dobře prováděl a bezpečně testoval, nicméně přibyly nadbytečné operace okolo - URL se musela vkopírovat do virtuálního stroje, přičemž někdy komunikace s virtuálním strojem selhávala a clipboard nešel použít. URL se tedy ukládala do souboru ve sdílené složce a ve virtuálním stroji se kopírovala zpátky do prohlížeče. Proces se podařilo částečně urychlit přes externí webový server, ke kterému měla přístup jak lokální stanice, tak virtuální stroj. Externí server přijal URL z MDM a pro virtuální stroj nabízel rozhraní, kde se URL sama rozklikla. (Na screenshotu předběžné implementace (výše) funkci zajišťoval automatický link "vm" jako *virtualmachine.*) Nicméně přestože výsledky analýzy ve virtuálním stroji byly dobré, byl totožný problém zjištěné údaje přenést bezpečně zpět na lokální stanici. Jednalo se tedy o kostrbaté nefunkční řešení.

4.4.2 Vlastní analyzér

Po neúspěšných pokusech jsme dospěli k tomu, že největší nadějí na úspěch je stavba vlastního analyzéru. Měl by zvládnout v nějakém svém virtuálním stroji pustit běžný prohlížeč, na který malware cílí, a proto se snáze projeví s větší pravděpodobností než při statické analýze zdrojového kódu, načteného přímo do analyzéru z *http*. Měl by umět komunikovat s tímto prohlížečem a přesně říci, na které URL se prohlížeč při navštívení URL dostal. A měl by mít API, s kterou by jednoduše dodával data do MDM a přebíral URL k analýze.

V případě, že bychom měli tento stroj, pomocnou vrstvu mezi MDM a uživatelem, mohl by implementovat i další automatizační požadavky, které práci s MDM urychlí a které by externí služby stěží mohly přinést:

- pamatovat si výsledky analýzy, aby se k nim mohl administrátor vrátit
- generovat škodlivá URL v CSV, abychom tím vytvořili seznam objevených škodlivých IP a adres (pro proprietární využití v projekt Turris)
- sám předpovídat škodlivá URL podle známých vzorů (např: načítání skriptu ".php" ze souboru na cizí doméně)
- umožnit adminstrátorovi rozhodnout, které domény, IP nebo URL se budou propříště automaticky považovat za škodlivé (protože jeden typ malware často zasahuje mnoho webových prezentací, administrátor by se nemusel ručně rozhodovat u každé domény, ale rovnou viděl, že je stránka nakažená)

Screenshot Analyzér by dále měl být schopen udělat screenshot, protože screenshot stránky přináší velmi cenné informace. Pokud je stránka útočným zásahem zcela zdemolovaná, administrátor to zjistí na první pohled a nemusí dál zkoumat. Experimentovali jsme s externími službami na automatické pořízení screenshotů, jmenovitě se službou url2png [footlink]. Nicméně pokud bychom chtěli zobrazit obrázek v analýze, narazili jsme na to, že služba je placená.

Navíc by bylo vhodné, aby screenshot odrážel výsledky běhu konkrétní analýzy, neboť nepřátelský skript může během 2 spuštění vrátit různé výsledky, a tak by měl být pořízen přímo analyzérem.

5 Implementace

V této kapitole se podíváme na realizace řešení navrhnutého v návrhové části. Popíšeme použité technologie, jeho architekturu a zajímavé algoritmy.

5.1 Analyzér

Virtuální stroj, kde běží server analyzéru, je napsán v jazyce Python3. Spouští bezpečně sandboxované Firefoxy, které se snaží simulovat běžného návštěvníka, na kterého může škodlivý kód zaútočit. Server dále obsluhuje databázi a nabízí dvě komunikační rozhraní - první dostupné přímo na adrese stroje přes *http* protokol, umožňuje spustit analýzu URL. Druhé rozhraní je určeno pro MDM, kde se výsledek analýzy integruje do původní stránky incidentu, ze které je analýza volána, a obohacuje ji o další ovládací prvky.

5.1.1 Sandboxovaný Firefox

Server může spouštět až 20 různých profilů Firefoxu. Ty jsou speciálně nastaveny, aby ignorovaly Google Safebrowsing a další bezpečnostní mechanismy prohlížeče. Každý profil má nainstalováno rozšíření, které umožňuje další analyzační funkce.

Přímý log z prohlížeče Firefox je spuštěn s parametrem, který ho přiměje vést log všech navštívených stránek. Ten má tu výhodu, že pokud jeho běh selže a program spadne, přesto máme jistotu, že získáme informace o běhu.

Doplněk do Firefoxu V rámci práce jsme naprogramovali vlastní rozšíření, které je v každém Firefoxu nainstalované ze společné složky. Rozšíření se stará o hlubší analýzu kódu v několika modulech.

Jakmile může, spustí doplněk špionážní modul, který má za cíl zjistit, které škodlivé příkazy stránka vykoná. Hned potom nastartuje paralelně modul traffic sniffer, který odposlouchává veškerou komunikaci. Potom pět vteřin počká, aby se škodlivé kódy stihly na stránce pohodlně vykonat. Od této chvíle čeká na zprávu, že se všechny stránky vykonaly a udělá screenshot, jak stránka dopadla. Čeká maximálně dalších pět vteřin, potom udělá screenshot a stránku zavře násilím, protože některé stránky jsou schopny načítat skripty donekonečna ve smyčce a průběh by neměl konec. Pokud se přesto něco stane a prohlížeč neskončí (například pokud některý malware porouchá prohlížeč natolik, že přestane reagovat), 25 vteřin po spuštění prohlížeče ho server ukončí násilím, i za cenu toho, že nedostane screenshot.

Modul: Traffic sniffer Modul Traffic sniffer se stará o to, aby nám neunikla žádná stránka, která se načte. Protože malware unáší uživatelův prohlížeč na finální destinaci často skrz několik vedlejších webových stránek, chceme mít přehled o tom, kudy všude prohlížeč šel a s jakým kódem se setkal. K odposlouchávání kódu používáme podobný kód jako rozšíření Firebug[Odv, 2008].

Velký problém byl, že se úvodní URL začala načítat ihned se startem prohlížeče, ještě předtím, než doplněk začal kód zachytávat. První URL tedy často prošla bez povšimnutí. Zajímavé je, že se to kvůli interním procesům Firefoxu nedělo vždy, občas Firefox zdržela služba Heartbeat, což je projekt telemetrie od Mozilly, která se vzorku uživatelů občas zeptá na spokojenost s využíváním prohlížeče. [ ]

Všimnul jsem si, že i v nástrojích vývojáře a v rozšíření Firebug, pokud chceme zobrazit status sítě, musíme poprvé znovu načíst stránku. Je tedy možné, že funkcionalita není přístupná ani programátorům, důkladně obeznámeným s vnitřní strukturou Firefoxu.

Měli jsme na výběr tedy ze dvou metod, jak první stránky přesto načíst. Buď pro první request využít vnitřní modul Firefoxu PageLoad, který ale nezachytí data tak spolehlivě jako odposlouchávání. Druhá možnost, kterou jsme zvolili, je mezistránka. Místo abychom startovali Firefox na URL http://example.com, startujeme ho na http://localhost/redirec V momentě, kdy se doplněk úspěšně načte, zjistí tuto speciální stránku a prohlížeč bezpečně přesměruje.

Modul: Spy Doplněk NoScript sice blokuje škodlivé skripty, ale současně ničí jejich práci. Nezjistíme s ním tak, co skripty měly v úmyslu udělat. Modul Spy nemá za cíl chránit uživatele a místo toho tak předefinuje přímo původní javascriptové funkce prohlížeče. Pokud kód například volá funkci *unescape*, zavolá se místo ní naše nahrazená *unescape*, která zalogguje parametr a vrátí výsledek stejný, jako originální *unescape.* Malware nepozná žádný rozdíl a provede se. Takto snadno dešifrujeme, jaký kód se přesně provádí.

Velké problémy jsme měli s implementováním narušené funkce *eval*, protože se nejedná o funkci v pravém slova smyslu, ale o jazykový konstruktu javascript, takže pracuje na nízké úrovni. Při testování jsme se totiž setkali se skriptem, který ve funkci *eval* volá jako parametr další konstrukci, funkci *arguments.callee.* Ta vrací odkaz na funkci z lokálního kontextu. V momentě, kdy jsme funkci *eval* nahradili špiónem a pak chtěli vrátit výsledek originální *eval*, *arguments.callee* nutně nabýval jiných hodnot. Místo funkce *arguments.callee* bychom mohli vložit funkce *arguments.callee.caller*, která jde v kontextu o úroveň výš, ale protože *eval* je jazykový konstrukt, řešení nefunguje. (Odzkoušeno shodně v implementaci ECMAScriptu interpreterem SpiderMonkey v aktuální verzi Firefoxu a v Chrome. Nakonec jsme v parametru *eval* přímo nahradili text *"arguments.callee"* textem "*arguments.callee.caller.name"*, což je metoda, kterou Mozzila, správce standardu, označuje za nestandardní a riskantní. Nicméně řešení funguje.

Funkce *document.write* je natolik interní, že se nám ji nepodařilo přepsat tak, abychom ji byli schopni znovu spustit. Abychom však neomezili malware v jeho běhu, simulujeme její funkčnost příkazem *document.getElementsByTagName("html")[0].innerHTML= str.* Máme naštěstí jistotu, že tag HTML existuje, neboť Firefox jej do DOMu doplňuje i na jinak zcela nevalidních stránkách.

Jakmile se provede nějaký podezřelý příkaz, modul odešle hlášení zpět do hlavní části doplňku. Ten zprávu ihned uloží do souboru, abychom o ni nepřišli v případě

pádu prohlížeče (který je stále pravděpodobný, protože škodlivý kód může mít nečekané následky). Protože zápis do souboru prohlížeč dělá pouze asynchronně a může mít i několik vteřin zpoždění, velký počet podezřelých funkcí, které se vykonávají těsně po sobě, má ten efekt, že je prohlížeč přestane stíhat zapisovat (ale nespadne). Přesto je to pro nás výhodnější zapisovat co nejrychleji, neboť máme jistotu, že se administrátorovi dostane zpráva o škodlivých příkazech - a ten si v případě většího zájmu může stránku zanalyzovat ručně.

Modul: Screenshot Do finální podoby stránky je injektován skript, který pošle údaje o konečných rozměrech stránky - výšku a šířku obrazovky. Malware podle injektáže eventuálně může přijít na to, že chci stránku upravit. Ale injektáž provádějí i běžná rozšíření, takže je nízká pravděpodobnost, že by malware poznal, že je v sandboxu. (Pravděpodobnost se dále snižuje zpožděnou injektáží.) Podle výšky a šířky vytvoříme objekt *canvas,* kam screenshot vložíme a z něj předáme do souboru ve formátu data-octet, v kódování base64. Nelze ukládat přímo png soubory, protože výstup implementace algoritmu base64, která je v doplňku dostupný, je možná vadný - lišil se od pozdějších výsledků, které base64 vracela v Pythonu. Necháme tedy screenshot posléze zpracovat Pythonem.

5.1.2 Server analyzéru

Server se primárně stará o spouštění sandboxovaných instancí prohlížeče. Dále zpracovává výstup z něj a v přijatelném formátu vypisuje pomocí rozhraní. Výsledky běhu prohlížeče se ukládají do datové struktury *crawl*, která je schopna pojmout všechny informace a odkazy na lokální soubory. Datovou strukturu *crawl* postupně obohacuje pět různých modulů, z nichž každý se stará o svou část logů. Struktura je potom cachována ve formátu *yaml,* který je čitelný člověkem.

Pokud je jedna URL analyzována vícekrát, server je schopen každou z nich uložit do zvláštní složky v cache adresáři, aby šly porovnat různé výsledky různých analýz téže stránky (a její posun k vyřešení malware nákazy).

Parser: NSPR log Zpracuje soubor logů, které zanechala daná instance Firefoxu, která je spouštěna s instrukcí důsledně loggovat události interního typu *nsHttp*, tedy všech možných navštívených domén. Loggování je natolik důsledné, že v logu nacházíme i informace o běžně neviditelných doménách - jako je například výše zmíněná služba Heartbeat.

Parser: Traffic Parser projde cache adresář a hledá všechny soubory, které vygeneroval modul doplňku Traffic sniffer podle předem domluveného formátu: Název souboru je ošetřený název konkrétního URL, přípona "tmp", na prvním řádku se vystkytuje plné znění URL (které by bylo nebezpečné uchovávat v názvu souboru) a *mimetype,* druh obsahu, jak ho Firefox vyhodnotil.

O tyto soubory obohatí objekt *crawl*. Při výpisu analýzy totiž chceme vidět jen podstatná data a ne všechny údaje, které máme k dispozici - ty jsou zobrazeny na vyžadání. Navíc chceme zdrojové soubory ukazovat s vyznačenou syntaxí a zkrácené o nepodstatné a neškodné objekty. K vyhledání zajímavých objektů (*iframe, script, img, embed...*) používáme knihovnu Pythonu *beautifulsoup,* která je sice pomalejší než například s Pythonem dodávaná *lxml,* ale zato si poradí i s různými nevalidními tagy [footnote : jak uvadeji autori, dle kreda této knihovny, ze "tuhle hroznou stranku jsem nepsal, ale musim ji parsovat"]. Syntaxi pak zvýrazníme s pomocí knihovny *pygmentize.* V případě HTML použijeme na obarvení defaultní třídu HTMLLexer, v případě skriptů jsme si však udělali situaci složitější. Protože javascriptové knihovny jsou dlouhé, chceme pro lepší orientaci zvýraznit funkce, které považujeme za potenciálně nebezpečné. K tomu jsme si napsali vlastní gramatiku jazyka, který vychází v *javascriptu,* třídu MdmaugJsLexer. Funkce jako *document.write* se vypisují obrovským červeným fontem, aby je administrátor viděl na první pohled. Jsme si vědomi toho, že výsledek nemůže podchytit všechny nebezpečné funkce, neboť javascript umožňuje jejich obfuskaci, a místo: "document.write" může být ve zdrojovém kódu napsáno jen: "a='doc'+'ument.write';".

(Analýza objektů typu QuickTime a Flashe by byla velmi složitá, naštěstí se jí nemusíme vůbec zabývat - protože už jen pouhá přítomnost těchto objektů na stránce je podezřelým signálem.)

Parser: Spy Parser projde soubory z modulu doplňky Spy podle podobně domluveného formátu. Jednotlivá chycená volání nebezpečných funkcí ošetří proti XSS a vrací objektu *crawl* k přímému výpisu jako formátovaný HTML kód. Pokud ovšem analýza ukáže velký počet podezřelých funkcí, anebo pokud jsou některé údaje o funkcích příliš dlouhé (typicky, když hacker volá *unescape* s veledlouhým parametrem, který dynamicky rozbaluje škodlivou funkci), vrátí jen 100 znaků pro maximálně 10 volání a přidá permanentní odkaz, pod kterým administrátor nalezne původní kód.

Parser: Metadata Obohacuje objekt *crawl* o další informace, které přímo nesouvisí s během instance prohlížeče. První z nich je geolokace. Ke každé doméně dohledá všechny IP adresy a k těm se pokouší zjistit geolokaci. Protože se instance Firefoxu během běhu často připojuje i k 20 stránkám a určení geolokace probíhá z externí služby, spustíme na každou doménu i na každý geodotaz vlastní vlákno. Přitom hlídáme, abychom negenerovali příliš mnoho vláken, což by vedlo k pádu celé analýzy - pokud je počet vláken větší než n, počkáme, až vlákna doběhnou, konsolidujeme je a teprve potom pustíme další várku. (S číslem n stále experimentujeme.)

Druhý typ metadat jsou údaje z databáze. Ke každé IP se přiřadí záznam, jaké doméně (doménám) patří, v rámci jaké analýzy jsme ji potkali a kdy a zda jsme ji vyhodnotili jako škodlivou, podezřelou, nebo důvěryhodnou. Objekt *crawl* ihned získá přehled o doménách, se kterými jsme se setkali v minulosti, a v analýze červeně podtrhává škodlivé domény, zatímco informace o důvěryhodných doménách sráží na minimum a řadí je dospodu.

Parser: Screenshot Aby mohl Firefox běžet, musí být k dispozici displej. Na virtuálním stroji však používáme jen virtuální displej, realizovaný knihovnou *xvbf.* Původně jsme chtěli screenshot získávat bez asistence Firefoxu, pouhým sejmutím aktuálního stavu virtuálního displeje, řešení z *xvbf* se však ukázalo jako příliš pomalé a zdržovalo běh skriptu. Implementovali jsme proto sejmutí screenshotu přímo ve Firefoxu, má to tu výhodu, že vidíme nejen rozlišení virtuální obrazovky, ale obraz celé plochy, kterou finální stránka v prohlížeči zabírá.

Jak jsme se ovšem zmiňovali výše, ve Firefoxu jsme narazili na problém s dešifrováním base64. Parser Screenshot tedy načte base64 kód, přeloží ho do PNG formátu a při té příležitosti obrázek zmenší na únosné rozměry náhledu.

5.1.3 Databáze

Pro účely ukládání dat používáme úložiště MySQL. Pro komunikaci s Pythonem jsme zvolili ORM databázovou vrstvu knihovny *peewee,* abychom mohli s objekty databáze zacházet jako se třídami Pythonu.

Původně bylo řešení implementováno pomocí *pymysql*, ta ale nezvládala sdílet zdroje mezi vlákny a vracela chybu *InterfaceError (0, '').* Protože komunita během tří let nenalezla řešení[5], přešli jsme na *peewee.*

5.2 Doplněk do MDM

Jako druhou programovací část jsme částečně automatizovali opakující se funkce v MDM. To jsme realizovali pomocí doplňku Firefoxu Greasemonkey, který umožňuje na daných stránkách spouštět uživatelský javascript.

Činnost Greasemonkey začíná hned na úvodní stránce. Přihlašování je realizováno pomocí služby MojeID. Po vyplnění uživatelského jména je uživatel na druhé stránce tupě vyzván, aby zmáčkl "ok". Oba tyto kroky jsou přeskočeny, uživatelské jméno se vyplní automaticky a tlačítko se zmáčkne. (Heslo služba neukládá, to je jednou za měsíc uživatel nucen napsat ručně.)

5.2.1 Hlavní stránka

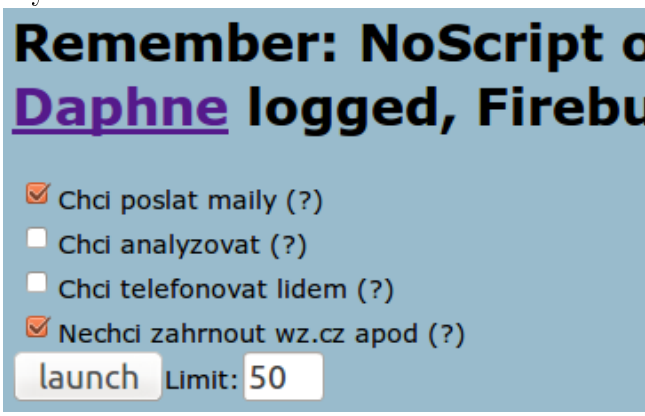
Do úvodní stránky je vložen *iframe,* který zajišťuje spojení s interním registrem domén. Stačí vyplnit heslo a kliknout na tlačítko "login" a skripty v Greasemonkey obstarají zbytek, spouští se totiž i v registru domén. Registr vložený v *iframe* s MDM komunikuje pomocí bezpečného API HTML pro posílání zpráv. Informace mohou téci skrz *iframe* v rámci dokumentu a není realizováno žádné internetové spojení, takže nehrozí únik informací. Elegantně jsme se tak dostali k datům z registru, aniž bychom ohrozili bezpečnost.

[5] http://stackoverflow.com/questions/6650940/interfaceerror-0

[greasemonkey - integrace registru a mdm]

5.2.2 Procházení incidentů

Javascriptová nadstavba dávkově projde desítky nahlášených domén naráz, za uživatele se sama prokliká do detailů a díky provázanosti s interním registrem domén si vytahá neveřejné e-maily vlastníků (všechny možné). Pokud nestojí v cestě žádná překážka, zvolí správnou šablonu e-mailu, odešle jej a je-li vše v pořádku, zavře tab prohlížeče. Z toho plyne, že nejdůležitější úlohu systému, okamžité informování vlastníků domén o hrozbách na jejich serveru, může převzít *cron*, automatický bot.



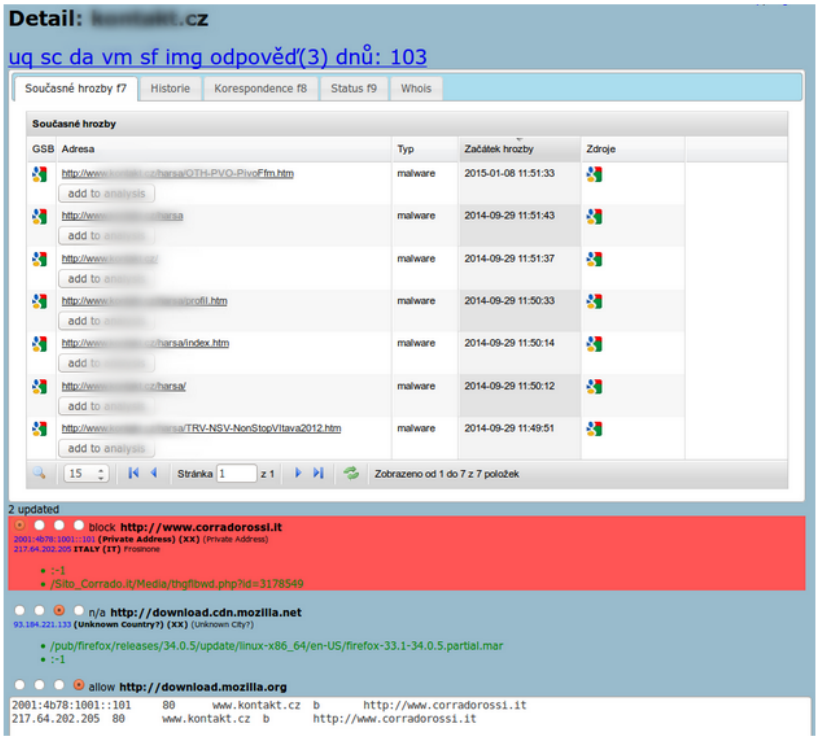
Toto menu přibylo na stránku s přehledem incidentů: incidentů]

V nadpisu jsou všechny důležité připomínky - pro běh Greasemonky je třeba mít zaplé rozšíření NoScript (kdyby se administrátor náhodou ukliknul a přešel na škodlivou doménu), je třeba být přihlášen v interním registru a deaktivovat rozšíření Firebug, které v kombinaci s během náročných uživatelských skriptů působí zamrznutí tabů.

Následuje krátké menu, které odpovídá požadavkům z návrhové části. Uživatelské skripty se zařídí podle toho, co chce administrátor dělat (zda poslat maily, analyzovat stránky nebo telefonovat lidem). Tlačítkem launch spustíme až 50 domén naráz. I když na serveru analyzátoru nemám k dispozici prostředky pro tolik Firefoxů, server nechá požadavky na analýzu čekat, až se zdroje uvolní (a instance Firefoxů doběhnou).

Se vzdáleným analyzačním serverem Greasemonkey plně komunikuje a předkládá výsledky. Snadno tak přijdeme i na kód, který je vysoce obfuskovaný - necháme působit samotný Firefox, aby se malware mohl projevit ve svém domácím prostředí, nerušen výskytem žádných bezpečnostních opatření.

Lidský uživatel má pak k dispozici přesný popis míst, které nevědomky nedobrovolně navštíví kdokoliv, kdo přistupuje na podezřelou URL. Několika pohyby šipkou pak vykoná hlasování o tom, která z URL je evidentně nebezpečná, kterou zaslouží zvýšenou ostražitost a které důvěřujeme natolik, že se s při další analýze nebudeme muset zabývat. Níže máme ukázku toho, jak vypadá napadená doména po analýze.



[obohhacenástránka s detailem incidendtu]

Úplně nahoře zůstal panel odkazů, který jsme popsali v návrhu, kdy URL můžeme postoupit ještě několika dalším externím službám (např. “uq” je urlQuery. . .). Nakonec je připojena informace, že v mailovém trackeru jsou 3 zprávy, přičemž od poslední uplynulo celých 103 dnů. Během této doby byla doména stále nakažená, protože její majitel nereagoval. Níže jsou taby opatřené klávesovými zkratkami, aby měl administrátor usnadněn pohyb. Každé nahlášené URL na doméně je opatřeno tlačítkem “add to analysis” pro případ, že bychom v rámci webu chtěli přiotestovat ještě další URL. (Např. když automatický nástroj napoprvé malware nenašel.) Lze je spustit všechny najednou, výsledky se nám sloučí s výsledky analýzy, která je pod tabulkou. V našem případě obsahuje tři weby. Hned první z nich je dle URL *corradorossi.it,* jasný případ jedné z mnoha domén, které jsou udržovány jen za účelem šíření malware - zpola smysl dávající název a podezřelé jméno PHP skriptu s číselným parametrem, takto posledních několik let vypadá nejčastější příklad malware odkazu. Další plus, kterým šetříme čas,

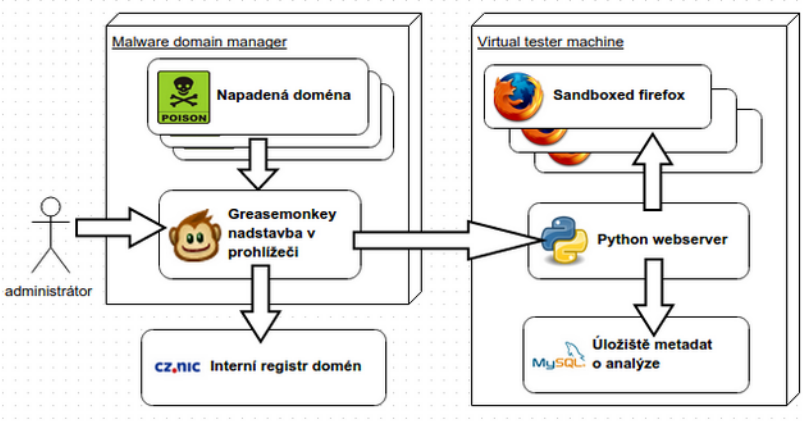
je dohledávání IP k doménám - každá doména má vypsány modře všechny adresy, ke kterým se vztahuje, a to jak IPv4, tak IPv6. Další doména patří regulérnímu downloadu mozilla.net a poslední je důvěryhodná mozilla.org, která se pro úplnost zařadila na konec seznamu.

Pod analýzou se neustále generuje tabulka, kde je možný CSV export pro projekt Turris.

Zpožděná analýza Domény nemusíme analyzovat rovnou. Jakmile se s nimi analyzátor setká, označí je. V momentě, kdy si uděláme čas na důkladné zkoumání podezřelých domén, využijeme rozšířené menu aplikace - položku "Malware domain kandidáti". Greasemonkey nám vygeneruje stránku se všemi doménami, které ještě vyžadují naše rozhodnutí.

[obohacené hlavní menu]

To administrátora osvobozuje od situací, kdy se v doménách, které analyzoval/neanalyzoval smutně orientuje pouze pomocí fialového zbarvení navštívených odkazů v prohlížeči. Jednou za určený čas můžeme provést také dávkový export CSV pro Turris další novou položkou v menu.



[přehled vylepšeného řešení]

6 Uživatelské testování

Tetování s uživatelem má za cíl zjistit několik věcí.

1. zda je návrh rozložení ovládacích prvků dobrý, nebo zmatečný
2. zda uživatel bude umět intuitivně využít novou funkcionalitu
3. zda se uživatel bude orientovat ve výsledcích analýzy

Očekávaný uživatel aplikace je kolega autora. Původní aplikaci MDM zná a dovede se v ní dobře orientovat. Proběhla dvě testování, v prvním jsme zkoušeli jenom 3. faktor, orientaci ve výsledcích analýzy, protože to je stěžejní část práce.

6.0.3 Průběh testování

Před prvním testováním jsem připravil 7 různých pokusných stránek a uživatel měl za cíl najít škodlivý kód. Nyní popíšeme, jak dlouho trvalo neškolenému uživateli najít škodlivý kód v pokusné stránce a kde byl umístěn.

1. 20 s, kód byl v tagu *script*, který unášel prohlížeč návštěvníka na cizí doménu *emercruz.com/vfjw7gtx.php?id=17007351*
2. 5 s, kód byl vložen v tagu *iframe*
3. 25 s, kód byl vložený ve skriptu jako v případě (1)
4. 10 s, škodlivý kód se na stránce nenacházel, protože doména byla zrušena
5. 30 s, kód byl v inline javascriptu, přičemž směřoval na obfuskovanou doménu *www.fotokorri.wz.cz.9080f5db02e72f04.d99q.cn*
6. 5 s, kód byl v iframe
7. 25 s, kód byl v neviditelném tagu img a směroval na IP (místo na doménu) *88.198.51.82/dev/cnt/cnt.php?d=gufex.cz*

Ve všech případech byl škodlivý kód správně určen. V případě testu (4) pro orientaci pomohlo, že analýza obsahovalo screenshot s jasnou informací o zrušení domény. Nedostatky jsme nalezli při identifikaci kódu v javascriptu, který trval uživateli výrazně déle.

Do druhého testování jsme proto upravili zacházení s javascriptem. Kód je teď dostupný pouze na vyžádání, administrátor musí kliknout na odkaz a formátovaný výstup se otevře v novém okně.

Druhé testování proběhlo přímo na aplikaci MDM. Uživatel se měl přihlásit do interního registru domén, u 3 nových incidentů zaslat majitelům domén maily, spustit analýzu náhodných domén a vyhodnotit ji.

Uživatel správně zvolil v menu, že chce posílat maily a vyhodnocovat analýzu, navolil číslo 3 a kliknul na tlačítko "launch". Maily se zaslaly samy a zůstaly taby s analýzou. Škodlivý kód na dvou z nich se nacházel v externím tagu *script*, který uživatel tentokrát bez problému našel. V třetím případě byl škodlivý kód vložen přímo do napadené stránky a používal funkci *eval,* kterou uživatel správně identifikoval jako škodlivou a hlasoval stránku zablokovat.

6.1 Testování bez uživatele

Testování aplikace bez uživatele má za cíl podrobit systém vysoké zátěži. Během testování jsme sledovali log analyzátoru a stav hostitelského počítače pomocí utility *htop.*

Ukázalo se, že běh konfigurovaného počtu 20 firefoxů stroj naráz nezvládne. Stroj začal házet výjimky ohledně alokace paměti (k dispozici má 1 GB RAM) a procesy Firefoxů zůstávaly viset. Původně jsme se domnívali, že když stroj zvládne Firefox s 20 taby, zvládne i 20 instancí, ale byla to mylná domněnka. V praxi jsme počet omezili na 3 Firefox a upravili server tak, aby v případě, že má více požadavků na analýzy než volných instancí, nechal požadavky viset tak dlouho, než se instance uvolní.

7 Diskuze

V této části se podíváme, jaké nové poznatky zprovoznění služby přineslo, zhodnotíme jeho přínos a nastíníme možný rozvoj.

7.1 Nové poznatky

Zajímavých výsledků jsem dostáhl s propíranou napadenou stránkou *www.starechovaci.cz* . Útočník tam využívá příkaz *eval* s krátkým argumentem v několika tisících iteracích. Poslední z nich nastavuje cookie a přesměrovává na škodlivou doménu s italskou koncovkou.

> eval: " function p09() {\r\n var static='ajax';\r\n var controller='index.php';\r\n
var p = document.createElement('iframe');\r\n\r\n p.src = 'http://www.dedoniturismo.it/cnt.php
p.style.position = 'absolute';\r\n p.style.color = '543';\r\n p.style.height
= '543px';\r\n p.style.width = '543px';\r\n p.style.left = '1000543';\r\n
p.style.top = '1000543';\r\n\r\n\r\n if (!document.getElementById('p')) {\r\n
document.write('<p id=\\'p\\' class=\\'p09\\' ></p>');\r\n document.getElementById('p').ap
}\r\n}\r\nfunction SetCookie(cookieName,cookieValue,nDays,path) {\r\n
var today = new Date();\r\n var expire = new Date();\r\n if (nDays==null
|| nDays==0) nDays=1;\r\n expire.setTime(today.getTime() + 3600000*24*nDays);\r\n
document.cookie = cookieName+\"=\"+escape(cookieValue)\r\n + \";ex-
pires=\" + expire.toGMTString() + ((path) ? \"; path=\" + path :
\"\");\r\n}\r\nfunction GetCookie(name) {\r\n var start = document.cookie.indexOf(
name + \"=\");\r\n var len = start + name.length + 1;\r\n if ((!start
) &&\r\n (name != document.cookie.substring(0, name.length)))\r\n
{\r\n return null;\r\n }\r\n if (start == -1) return null;\r\n var end =
document.cookie.indexOf(\";\", len);\r\n if (end == -1) end = doc-
ument.cookie.length;\r\n return unescape(document.cookie.substring(len,
end));\r\n}\r\nif (navigator.cookieEnabled)\r\n{\r\nif(GetCookie('visited_uq')==55){}else{Se
'55', '1', '/');\r\n\r\np09();\r\n}\r\n}"

Na screenshotu vidíme seznam několika prvních *evalů.* Na konci je šipka s odkazem, který vede na plné znění škodlivého kódu.

Mnoho zkušeností jsem nabyl také s vypínáním cache v prohlížeči, který měl tendence opakovaně ukládat mezivýsledky napadených stránek. Analýza pak nepřinesla očekávané výsledky. Seznam nutných preferencí, které je třeba nastavit každé používané instanci sandboxovaného Firefoxu, uvádím v dokumentaci zdrojového kódu, spolu s tipem, jak soubor prefs.js, kde Firefox preference z meta stránky *about:config,* ukládá, efektivně nakopírovat do všech ostatních instancí, aniž bychom je museli spouštět.

7.2 Přínos a další rozvoj

Služba významně přispěla k práci se systémem MDM. Níže vidíme část analýzy stránky *fotokorri.wz.cz* . Server správně identifikoval škodlivý kód stejně jako služby urlQuery a Sucuri. Níže na screenshotu jsou URL s klíčovým slovem *allow* jejichž výpis je zkrácen, protože se jedná o dříve potkané důvěryhodné domény.

[fanalýza fotokorri]

V současnosti je ve službě MDM kolem 400 zastaralých domén, jejichž majitelé nereagují a které je třeba analyzovat, zda kód stále obsahují, nebo zda zůstaly nedopatřením v seznamu veřejných zdrojů škodlivých domén. Díky tomuto projektu bude možno tento počet v příštích měsících významně snížit.

V cca jednom z deseti případů se stává, že analýza selže a malware není správně detekován. K zlepšení detekce můžeme experimentovat například se změnou hlavičky *user-agent,* abychom přiměli projevit se i malware, který čeká na jiné prohlížeče.

Do budoucna plánujeme otevřít analyzér jako veřejnou službu - nyní běží v interní VPN společnosti a zvenku se k rozhraní nikdo nedostane.

8 Shrnutí

V práci jsme probrali současné typy škodlivého kódu, zvláště jsme se zaměřili na phishing a malware na stránkách. Probrali jsme, jak se jednotlivé typy malwaru na stránky dostanou, co dělají s návštěvníky a co je jich cílem, a možnosti, jak se malwaru bránit. V další fázi jsme se zabývali rozvojem systému MDM, který slouží ke správě škodlivých domén. Práce s ním má být rychlejší a také má zahrnovat možnost analýzy URL adres, které jsou z výskytu škodlivého kódu podezřelé.

Zrychlení MDM jsme implementovali pomocí uživatelských javascriptů, které obohacují jednotlivé stránky o novou funkcionalitu. Zdroje uživatelských skriptů jsou uloženy na interním sdíleném disku, kde probíhá vývoj, takže změny se ihned projeví všem administrátorům, kteří MDMAUG používají. Analyzér pak běží pod názvem MDMAUG (augmentované MDM) v interní síti CZ.NIC. Zdrojové kódy lze stáhnout na stránce projektu[6].

[6] http://web.edvard.cz/mdmaug/

References

[Lis,] The list of malware types.

[Odv, 2008] (2008). Nsitraceablechannel, intercept http traffic. In *Software is hard*.

[Pan, 2014] (2014). Twenty percent of all malware appeared in 2013. In *Panda security*.

[Kas, 2015] (2015). Prank programs, altruistic malware and stoned viruses: Kaspersky lab remembers 'benign' malware.

[Slo, 2015] (2015). Slovensko zažilo jeden z najväčších internetových útokov v histórii. In *HN style*.

[Bašta, 2014] Bašta, P. (2014). 787, 618, 302, 67, 18, aneb statistiky jednoho „bláznivého oběda".

[CZ.NIC, 2012] CZ.NIC (2012). Zpráva o činnosti csirt.cz (národního csirt Čr) za rok 2012.

[Dube et al., 2010] Dube, T., Raines, R., Peterson, G., Bauer, K., Grimaila, M., and Rogers, S. (2010). Malware type recognition and cyber situational awareness. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 938–943.

[Husovec, 2014] Husovec, M. (c2014). *Zodpovednosť na internete*. CZ.NIC, Praha.

[Kishore et al., 2014] Kishore, K., Mallesh, M., Jyostna, G., Eswari, P., and Sarma, S. (2014). Browser js guard: Detects and defends against malicious javascript injection based drive by download attacks. In *Applications of Digital Information and Web Technologies (ICADIWT), 2014 Fifth International Conference on the*, pages 92–100.

[Kolbitsch et al., 2012] Kolbitsch, C., Livshits, B., Zorn, B., and Seifert, C. (2012). Rozzle: De-cloaking internet malware. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 443–457.

[Kümmel, 2011] Kümmel, R. (2011). *XSS*. Tigris, Zlín.

[Mohebzada et al., 2012] Mohebzada, J., El Zarka, A., Bhojani, A., and Darwish, A. (2012). Phishing in a university community: Two large scale phishing experiments. In *Innovations in Information Technology (IIT), 2012 International Conference on*, pages 249–254.

[Phakoontod and Limthanmaphon, 2012] Phakoontod, C. and Limthanmaphon (2012). Malicious web page detection based on feature classification. In *Computing and Convergence Technology (ICCCT), 2012 7th*, pages 66 – 71.

[Pilgrim, 2010] Pilgrim, M. (c2010). *Ponořme se do Python(u) 3*. Cz.Nic, Praha.

[Sachin and Chiplunkar, 2012] Sachin, V. and Chiplunkar, N. (2012). Surfguard javascript instrumentation-based defense against drive-by downloads. In *Recent Advances in Computing and Software Systems (RACSS), 2012 International Conference on*, pages 267–272.

[Scambray and Shema, 2002] Scambray, J. and Shema, M. (c2002). *Hacking exposed*. McGraw-Hill/Osborne, New York.

[Ďurechová,] Ďurechová, K. Attacks on the web honeypot.

[Ďurechová and Duračinská, 2014] Ďurechová, K. and Duračinská, Z. (2014). Čo prinieslo skenovanie heartbleed bugu.

ČVUT, FEL

Detekce škodlivého kódu

ve webových stránkách

Edvard Rejthar

Studijní program: Otevřená informatika

Vedoucí práce: Ing. Michal Medvecký

Abstrakt

Následující práce si klade za cíl zpracovat rešerši škodlivého kódu, který útočníci vkládají do napadených webových prezentací. Posléze navrhneme a implementujeme program, který bude schopný vyhledat ve skriptech webu podezřelý kód a vypsat škodlivé IP domény, na které je návštěvníkův prohlížeč škodlivým kóden unášen. Program implementujeme do existující aplikaci Malicious Domain Manager, která slouží ke správě údajů o napadených doménách a otestujeme.

Klíčová slova: malware, hrozba, Firefox, automatizace, síť, hacker, doplňky

Abstract

Main subject of this work is to analyze the types of malware code that attackers include into hacked websites. Then we design and realize program capable of suspicious code detection in website scripts, that lists the IP addresses of malicious domains visitor's browser is redirected to. We implement the program in the current application Malicious Domain Manager that serves as an incident handling platform.

Key words: malware, threat, Firefox, automation, network, hacker, addons

Prohlášení

Tímto prohlašuji, že jsem práci vypracoval samostatně s použitím odborné literatury a pramenů uvedených v seznamu, který je nedílnou součástí této práce. Zároveň prohlašuji, že nemám námitek proti použití tohoto školního díla ve smyslu § 60 zákona č. 121/2000 Sb., o autorských právech a právech souvisejících, ve smyslu pozdějších znění tohoto zákona.