Česká společnost uživatelů otevřených systémů EurOpen.CZ
Czech Open System Users' Group
www.europen.cz

39. konference

Sborník příspěvků

klášter Želiv
2.–5. října 2011

Sborník příspěvků z 39. konference EurOpen.CZ, 2.–5. října 2011

Plzeň 2011. První vydání.
Editoři: Vladimír Rudolf, Petr Švenda, Václav Matyáš
Sazba a grafická úprava: Ing. Miloš Brejcha – Vydavatelský servis, Plzeň
e-mail: servis@vydavatelskyservis.cz

Tisk: Typos, tiskařské závody, s. r. o.
Podnikatelská 1160/14, Plzeň

Příspěvky neprošly redakční ani jazykovou úpravou.

ISBN 978-80-86583-22-8

Obsah

Tutoriál: Programování kryptografických čipových karet na platformě JavaCard a jejich praktické použití v aplikacích

Petr Švenda

E-mail: svenda@fi.muni.cz

Abstrakt

V rámci tutoriálu každý účastník obdrží moderní programovatelnou multi-aplikační kryptografickou čipovou kartu (Gemalto TOP IM GX4 + Mifare Classic 1k) s platformou JavaCard (podporované algoritmy 3DES, AES, RSA 2 048bit, ...) a s pomocí předchystaného vývojového prostředí zvládne proces vývoje elementárního appletu, jeho nahrání na kartu a tvorbu jednoduché aplikace, která s kartou na straně PC bude komunikovat. Ve druhé části si na kartu nahrajeme několik existujících appletů využitelných pro uložení privátního podepisovacího klíče na čipovou kartu pro program PGP/GPG, vytvoření karty nabízející úložiště dle standardu PKCS#11 nebo PKCS#15 (použitelné pro širokou řadu aplikací) nebo aplikaci umožňující tvorbu klonu elektronického pasu. Probrány budou jak vývojové, tak i bezpečnostní aspekty používání programovatelných čipových karet.

Formáty pro zaručený elektronický podpis

Libor Dostálek

E-mail: dostalek@prf.jcu.cz

Bezpečnost dokumentů je v první řadě věcí kulturní tradice. Stačí si uvědomit, kolika snadno padělatelným dokumentům věříme. Klasickým případem je lékařský předpis. Je to dokument prakticky bez jakýchkoliv ochranných prvků. Razítko lékaře nám bez jakýchkoliv problémů běžně zhotoví. A podpis lékaře – ten se proti žádnému podpisovému vzoru nekontroluje.

Ano, v některých zemích se zavádí tzv. elektronický lékařský předpis, který je již zabezpečený. Ale jeho zavedení je spíše důsledkem IT loby než opravdové potřeby. Potřeba by vznikla v okamžiku, kdyby defraudace vzniklé padělanými recepty převýšily náklady na zavedení bezpečného elektronického receptu. Pokud bude elektronický recept zaveden, tak důvodem spíše bude sledování nadbytečného předepisování léků než nedostatečná bezpečnost papírového receptu.

Doba listin

V případě faktur, které díky tomu, že slouží jako daňové doklady, je situace ještě zajímavější. V případě listinných faktur je zákon velice benevolentní. Razítko a podpis na faktuře je spíše náš národní rituál než že by to bylo zákonem požadováno. Jedině díky tomu, že finanční úřad může zpochybňovat cokoliv (existuje zde presumpce viny), tak ten podpis a to razítko na fakturu každý raději vloží, aby úřad dostal, co jeho jest. Velcí výstavci faktur (např. telekomunikační operátoři, vodovodní sítě apod.) by se urazítkovali, tak aspoň faktury tisknout na hlavičkovém papíře. Hlavičkový papír je tak vlastně považován za ochranný prvek faktury.

Proč v případě faktury se na jejím podpisu (resp. indicií pravosti) příliš nebazíruje? Odpověď je velice jednoduchá. Faktura vždy existuje ve dvojím vyhodnocení a každá z těchto kopii si žije svým nezávislým vlastním životem. Jedna je u dodavatele a druhá u odběratele. Finanční úřad má přístup k oběma stranám. Bezesporu je příliš nákladné zmanipulovat obě vyhotovení (a tím i účetnictví) obou stran.

V případě listinných dokumentů máme řadu indicií pravosti dokumentu:

- podpis dokumentu, který může být srovnáván s podpisovým vzorem;

- razítko nebo pečeť;
- pravost nosiče (např. papíru) listiny. Že se jedná zejména o stáří nosiče, ochranné prvky (logo), technologii výroby (dokument je datován v době, kdy daná technologie výroby nebyla známa);
- existence více kopií uložených na různých místech (viz faktury);
- historické důkazy.

Historické důkazy se opírají o historická fakta uvedená v dokumentu, která mohou být v rozporu s informacemi uvedenými v dokumentu. Např. na fotografii je vidět sousední budova, která byla postavena o 100 let později, než je datován dokument. Nebo naopak je tam budova, která tam stála jen krátce a zrovna v době datace dokumentu.

Doba příchodu počítačů

Doba počítačů nejprve nahradila psací stroje. Tato doba totiž přinesla tlačítko „Backspace“. Zavedení tohoto tlačítka je podle mne srovnatelné s nástupem GSM nebo objevem parního stroje. Výsledkem „na počítači pořízeného“ dokumentu je na tiskárně počítače vytisknutý listinný dokument.

Na konci každého měsíce musím vytvořit knihu jízd a k ní přiložit vytištěné účtenky o čerpání pohodných hmot. Obzvláště v letních měsících se mi stává, že tisk na nich nevydrží ani měsíc. Vím, že se jedná o tisk z termotiskáren a listiny orgánů veřejné moci se zpravidla tisknou na laserových tiskárnách. Otázkou ale je jak, za jak dlouho se tento tisk změní zpět na prášek, ze kterého se zrodil (zrodil se z toneru). Ale ani tak nikdo nezabrání, aby stavební úřad neoblíbenému sousedovi nevytiskl stavební povolení na termotiskárně.

Císař zavedl pořádek

Císař zavedl do „životního cyklu dokumentů“ řád. Zavedl nám v úřadech podatelny a spisovny. Zavedl nám Spisové a skartační řády, Schvalovací řády, Evidenci pečetidel a razítek atd. U nás všechny tyto výdobytky považujeme za samozřejmost. Vždy jsem překvapen, když dorazím do země, která nebyla součástí Mocnářství, že:

- korespondence úřadu neprochází jednotným bodem – podatelnou (resp. výpravnou), ale přímo konkrétnímu úředníkovi;
- neexistuje Spisový a skartační řád, specifikující jak dlouho se mají dokumenty uchovávat;

- nikdo neeviduje, jaká úřad používá razítka;
- nikdo nespecifikuje, jaké dokumenty může kdo podepsat;
- neexistuje ustálená forma úředních dokumentů.

Zpočátku jsem si říkal, že to u nás není možné, že jsme vyspělou zemí, ale pak jsem se zastyděl. Už, už vás slyším namítat: „To u nás není možné“. Ale ruku na srdce. Opravdu:

- Vaši úředníci nekomunikují elektronickou poštou až příliš přímo (bez evidence)?
- Nenakupujete stále větší a větší disky pro vaše IT, protože nikdo neřeší, že by měly některé elektronické dokumenty projít procesem skartace (vždyť jsou dnes ty disky tak laciné)?
- Opravdu eviduje kvalifikované certifikáty (resp. systémové certifikáty), kterými elektronicky podepisujete vaše dokumenty?
- Opravdu máte doplněn Spisový a skartační řád o elektronické dokumenty?
- Opravdu máte doplněn Schvalovací řád o schvalování elektronických dokumentů?
- Domníváte se, že e-mail a dokumenty systému Datových schránek jsou správnou formou úředních dokumentů? Je elektronický podpis takovýchto zpráv (nikoliv samotných dokumentů) tím pravým ořechovým?

Když tak tu současnu elektronickou komunikaci našich úřadu z povzdálí sledují, tak mám chuť zvolat: Vraťte nám Františka Josefa!

Elektronický podpis

A to už jsme dorazili do současnosti. Dokumenty elektronicky podepisujeme a vkládáme do elektronických obálek. Směrnice 1999/93/EC Evropského parlamentu a rady nám zavádí elektronický podpis: *‚elektronickým podpisem‘ se rozumí data v elektronické podobě, která jsou připojena k jiným elektronickým datům nebo jsou s nimi logicky spojena a která slouží jako metoda autentizace.*

Definuje zaručený elektronický podpis (anglicky *Advanced Electronic Signature* – zkratkou AdES): *„zaručeným elektronickým podpisem“ se rozumí elektronický podpis, který splňuje tyto požadavky*:

a) *je jednoznačně spojen s podepisující osobou;*

b) *umožňuje identifikovat podepisující osobu;*

c) *je vytvořen s využitím prostředků, které podepisující osoba může mít plně pod svou kontrolou;*

d) *je spojen s daty, ke kterým se vztahuje tak, aby bylo možno zjistit jakoukoliv následnou změnu těchto dat;*

A dále: *Členské státy zajistí, aby zaručené elektronické podpisy (tj. AdES) založené na kvalifikovaných certifikátech a vytvořené pomocí prostředků pro bezpečné vytváření podpisu*:

a) *splňovaly právní požadavky na podpis ve vztahu k datům v elektronické podobě stejně, jako vlastnoruční podpisy splňují tyto požadavky ve vztahu k datům na papíře; a*

b) *byly přijímány jako důkazy v soudním řízení.*

A k tomu na druhou stranu dodává: *„Členské státy zajistí, aby elektronickým podpisům nebyla odpírána právní účinnost a aby nebyly odmítány jako důkazy v soudním řízení pouze z toho důvodu, že*:

- *jsou v elektronické podobě, nebo*
- ***nejsou založeny na kvalifikovaném certifikátu**, nebo*
- *nejsou založeny na kvalifikovaném certifikátu vydaném akreditovaným poskytovatelem certifikačních služeb, nebo*
- *nejsou vytvořeny pomocí prostředku pro bezpečné vytváření podpisu.“*

Elektronický podpis je Evropskou specialitou

Když k nám dorazili kolegové ze vzdálené země Papua-Nová Guinea, tak po několika dnech byli velice rozmrzelí. Nakonec to z nich vylezlo. Nechápali, proč jsou voděni po nějakých stavbách (historických památkách). Hostitelé nevěděli, co by jim ukázali. Papuánci to vysvětlili. Kdesi zahlédli v zahrádkářské kolonii exotické rostliny. Takové divné rostliny jako je růžičková kapusta nebo kedlubna totiž nikdy neviděli. Tam nás zaveďte...

Pokud nechápete, jak to souvisí s elektronickým podpisem, tak např. z hlediska Američanů je elektronický podpis něco jako kedlubna pro Papuánce. Nedávno se mi dostal do ruky dokument *The AdES family of standards: CAdES, XAdES, and PAdES, Implementation guidance for using electronic signatures in the European Union* firmy Adobe, kde se píše:

„This white paper is a guide for choosing and deploying electronic signature technology in the European Union (EU). It will help business managers, system integrators, and partners understand the choices available for implementing electronic signature applications that can be readily deployed and are assured to meet the requirements of the European Commission (EC) Electronic Signature Directive 1999/93/EC (hereafter referred to simply as ‚the Directive'). ... ***In Europe, electronic signatures*** *enable non-repudiation from a legal perspective and under some strict circumstances can be legally equivalent to handwritten signatures. For transactions, electronic signatures can provide greater efficiency and faster turnaround times by eliminating costly paper printing and mailing delays, making it possible to reduce errors associated with rekeying of information, and improving convenience for end users. They also complement regulatory compliance and long-term document retention."*

Myslím si, že naši Papuánci po návratu svým krajanům obdobně popisovali co to je ta kedlubna: „... In Europe ‚kedlubna' ... "

Elektronický podpis je přece tak jednoduché vytvářet

Po prvním rychlokurzu o elektronickém podpisu začnou iniciativní vyškolenci rozesílat elektronicky podepsané emaily, protože jsou pyšní, na to, že pochopili jak je to jednoduché. Avšak na rychlokurzu jim zpravidla opomněli vysvětlit, že certifikát mají vydaný testovací certifikační autoritou nebo v lepším případě od nějaké interní firemní certifikační autority. Adresátům mimo jejich firmu se pak jeví tento podpis nedůvěryhodným a tak ihned ví oč běží. Já občas dostávám od `cic@csas.cz` „nedůvěryhodné" elektronicky podepsané maily certifikátem vydaným vydavatelem:
CN = CSEROOT, OU = Sprava PKI, O = Ceska sporitelna a.s., C = CZ

Musím po pravdě přiznat, že první takový email mne opravdu vylekal, přemýšlel jsem totiž, že se nějaký spammer vydává za Českou spořitelnu, ale nemohl jsem pochopit jak mi oznámením, že v Českých Budějovicích zprovoznili nový bankomat, chce ublížit. Přitom Česká spořitelna pro elektronické bankovnictví používá certifikáty Verisignu nebo ICA, které by se mi jevily jako důvěryhodné.

Co v zákonu o elektronickém podpisu není

Evropská směrnice 1999/93/EC i náš zákon 227/2000 Sb. o elektronickém podpisu řeší problematiku vytvoření elektronického podpisu dokumentu. Ale taktně mlčí o tom co s ním. Jak jej dlouhodobě uchovávat.

To je právní pohled, ale v dalším odstavci pochopíme, že EU nás na holičkách nenechala, spíše sami si to kazíme.

Formáty pro zaručený elektronický podpis

Evropská směrnice 1999/93/ES [7] zavádí legislativní pojem zaručený elektronický podpis (anglicky *Advanced Electronic Signature* – zkratkou AdES). ETSI (The European Telecommunications Standards Institute) pak vydala řadu norem, které specifikují mj. standardy pro formáty zaručených elektronických podpis (AdES). Tyto standardy definují formát zaručeného podpisu, a to tak, aby elektronický podpis mohl být udržován dlouhodobě platným.

V češtině došlo k překladatelskému problému. Díky několikanásobnému překladu z angličtiny do češtiny a zpět termín *Advanced Electronic Signature* jednou překládáme jako zaručený elektronický podpis a podruhé jako rozšířený elektronický podpis. A najednou máme dva typy podpisů – dokonce vznikají bouřlivé diskuse o tom, jestli pro zaručený elektronický podpis máme využívat rozšířené elektronické podpisy nebo ne.

Skutečnost je ale taková, že máme legislativní rámec zaručeného elektronického podpisu [7] a pak technické standardy pro formáty zaručeného elektronického podpisu (ETSI). Nikoliv tedy, že by směrnice 1999/93/ES zaváděla zaručený elektronický podpis a ETSI specifikovala formáty pro rozšířený elektronický podpis. Snadno si to čtenář ověří, když otevře kteroukoliv z dále zmíněných norem ETSI.

ETSI nám standardizoval tři formáty pro zaručený podpis (AdES): CAdES, XAdES a PAdES. Tyto tři standard liší formátem dat, která podepisují:

- CAdES [1] je určen pro podpis obecných dat. Jedná se o rozšíření legendárního standardu *Cryptographic Message Syntax* – PKCS #7 [2], který byl později přejat jako RFC a dále rozpracováván jako CMS [3]. Z řetězce CMS pochází i počáteční písmeno „C“ ve zkratce CAdES.

- XAdES [5] pro podpis XML dokumentů. Formát vychází ze standardu [6].

- PAdES [4] nezavádí nový formát podpisu, ale promyšleně kombinuje formáty CAdES a XAdES pro interní podpis PDF dokumentů.

Formáty CMS [3] a DSIG [6] trpí některými nedostatky. Standardy CAdES a XAdES realizují protiopatření proti těmto nedostatkům a přidávají mechanismy potřebné pro podporu zaručeného podpisu, kterým je např. Politika elektronického podpisu. Nyní se zamyslíme nad některými nedostatky CMS a DSIG a zmíníme příslušná protiopatření.

Oprávněnost podpisu

Nevím, jestli slovo „oprávněnost“ je tím nejvýstižnějším slovem. Prakticky se jedná o to, že můžete vytvořit výpověď svému kolegovi a dokonce ji podepsat svým AdES a je to jen žert (takový žert ale může být považován za hrubé porušení pracovní kázně!), protože nejste oprávněn takovéto listiny podepisovat. Žert by to nebyl v případě, kdybyste byl ve Schvalovacím řádu vašeho úřadu uveden jako osoba oprávněná podepisovat výpovědi.

Oprávněnost podpisu se řeší v případě standardů AdES tzv. politikou elektronického podpisu.

Politika elektronického podpisu řeší ještě jiný problém. Zaručený podpis může být velice variabilní. Když jej vytváříte, tak si kladete otázky: Vytvořit externí nebo interní podpis? Jaké podepisované a jaké nepodepisované atributy by podpis měl obsahovat? Mám do podpisu vůbec přidávat „podpisový certifikát“? Mám do podpisu přidávat časová razítka?

Pokud mám elektronický podpis ověřit, tak je to ještě horší! Co mám v podpisu ověřovat a které nepodepisované atributy jsou nevýznamné? S jakými certifikačními politikami jsou pro mne certifikáty vůbec akceptovatelné? Atd.

Tvůrci software některé zmíněné atributy implementují a jiné nikoliv. Horší je to při ověřování podpisu, protože v podpisu se může objevit (resp. chybět) některý atribut a ověřování selže i když by selhat dle normy nemělo.

Důsledkem použití politiky elektronického podpisu mělo být, že pokud podpis vytvořím pod konkrétní politikou podpisu, pak pokud jej budou pod touž politikou ověřovat různé nezávislé strany, dojdou ke stejnému výsledku. Podpisy obsahující podepisovaný atribut Politika elektronického podpisu se označují jako EPES (*Explicit Policy-based Electronic Signatures*).

Před cca pěti lety se zdálo, že si nikdo nedokáže představit zaručený podpis, který by nebyl alespoň EPES. Realita je jiná EPES většina software neimplementoval a pokud jej implementovala tak jen formálně. Důvodů je několik:

- Implementace je nákladná.
- Zaručený podpis a zejména politika podpisu je evropské specifikum. Evropský trh je příliš malým na to, aby se nákladná investice vyplatila.
- Politika podpisu uspokojivě nevyřešila otázky typu: jestli podpis pod dokument připojila osoba oprávněná dokument podepsat, jestli dokument obsahuje všechny podpisy atd.

Autoři standardu CAdES, který byl prvním standardem z řady AdES, nejprve nechtěli připustit jako CAdES podpis neobsahující podepisovaný atribut politika elektronického podpisu, tj. vyžadovali jako nejjednodušší formát CAdES-EPES. Potíž byla v tom, že do té doby již existovalo velké množství podepsaných dokumentů, které tento podepisovaný atribut neměly a tak by technicky nemohly

bát AdES. Autoři se bránili tím, že když povolí AdES bez tohoto atributu (později označovaný jako AdES – *Basic Electronic Signature* – AdES-BES), tak celý mechanismus spadne pod stůl. Nakonec byli udoláni ale ukázalo se, že více či méně měli pravdu.

Platnost AdES v čase

Platnost AdES v čase se řeší tím, že k podpisu BES nebo EPES přidáme nepodepisovaný atributu „Časové razítko z elektronického podpisu“, vznikne tak podpis *Electronic Signature with Time* – AdES-T. Časové razítko nám konzervuje elektronický podpis v čase – slouží jako důkaz, že podpis existoval v čase, kdy data k jeho ověření (např. certifikáty) byly platné. Mělo by se k podpisu dodat v době, kdy je podpis platný, tj. co nejrychleji.

Jenže časové razítko má dobu platnosti omezenu dobou platnosti certifikátu autority pro vydávání časových razítek, a tak pokud chceme udržovat podpis platným i nadále, tak musíme podpis „časově přerazítkovávat“. Podpisy s dalšími razítky se označují jako *Archival Electronic Signature* – AdES-A.

Zatímco používání podpisu AdES-T nikdo nezpochybňuje, tak na neustále „přerazítkovávání“ formátu *AdES-A se mnozí dívají skrz prsty. Mají proto dva argumenty:

1. Případ elektronická faktura (daňový doklad). Dle zákona se faktura udržuje 10 let. Pakliže elektronický podpis pod elektronickou fakturou opatříme podpisem AdES-T, pak jej bez problémů budeme moci ověřovat 3–5 let. Jenže pravděpodobnost, že finanční úřad bude po více jak třech letech šetřit faktury není velká. Navíc, že by finanční úřad zrovna zpochybňoval na faktuře elektronický podpis je o to menší (finanční úřad zpravidla využívá důkaz „existence dvou faktur“), takže náklady na případné přerazítkovávání podpisů faktur se nemohou vrátit.

2. Případ trvale archivovaných dokumentů. Po 100 letech by dokument musel být doplněn o 20–30 časových razítek což je opravdu až legrační.

Zákon o archivnictví a spisové službě

Problém uchovávání elektronicky podepsaných dokumentů řeší Zákon o archivnictví a spisové službě (zákon č. 499/2004 Sb., o archivnictví a spisové službě a o změně některých zákonů, jak vyplývá z pozdějších změn). § 69a tohoto zákona naši problematiku vcelku uspokojivě řeší. Já osobně bych tento paragraf interpretoval asi následovně:

Měli bychom si vybudovat nějaké úložiště dokumentů, které bude dokumenty udržovat *„postupem zaručujícím věrohodnost původu dokumentu, neporušitelnost jeho obsahu a čitelnost dokumentu, a to včetně údajů prokazujících existenci dokumentu v digitální podobě v čase.“* Na což máme technické standardy (např. standard OAIS). Před ukládáním dokumentů do tohoto úložiště nesmíme zapomenout ověřit platnost podpisů i časových razítek, čemž budeme v metadatech udržovat záznam.

§ ***69a*** **Zákona o archivnictví a spisové službě**

(2) Je-li doručený dokument v digitální podobě opatřen uznávaným elektronickým podpisem, elektronickou značkou nebo kvalifikovaným časovým razítkem, určený původce

(a) ověří platnost uznávaného elektronického podpisu, elektronické značky nebo kvalifikovaného časového razítka a platnost kvalifikovaného certifikátu nebo kvalifikovaného systémového certifikátu,

(b) zaznamená údaje o výsledku ověření podle písmena a) a uchová je spolu s doručeným dokumentem digitální podobě.

(3) Uchovávání dokumentu v digitální podobě provádí určený původce postupem zaručujícím věrohodnost původu dokumentu, neporušitelnost jeho obsahu a čitelnost dokumentu, a to včetně údajů prokazujících existenci dokumentu v digitální podobě v čase. Tyto vlastnosti musí být zachovány po dobu skartační lhůty dokumentu. Je-li potřeba zachování věrohodnosti původu dokumentu kratší než skartační lhůta dokumentu, uvede to určený původce ve svém spisovém a skartačním plánu.

(8) Neprokáže-li se opak, dokument v digitální podobě se považuje za pravý, byl-li podepsán platným uznávaným elektronickým podpisem nebo označen platnou elektronickou značkou osoby, která k tomu byla v okamžiku podepsání nebo označení oprávněna...

Soumrak elektronického podpisu?

Na první pohled by se mohlo zdát, že AdES v kombinaci se zařízením pro bezpečné vytváření elektronického podpisu je ideálním prostředkem nahrazujícím rukou psaný podpis. Problém je v tom, že v případě úřadů a firem je to pravda. Avšak pro běžné občany je elektronický podpis přímo noční můra. Příliš nechápou jeho princip a navíc pro ně není uživatelsky moc přívětivý. Pokud mají možnost se mu vyhnout, pak se mu vyhnou.

SMS podpis

Pro běžné občany je pochopitelné a přívětivé využívat jednorázová hesla zasílaná přes SMS („SMS podpis“). Je to pro ně jasné asi tak, jako je jasné vytáhnout razítko z šuplíku a otisknout jej nad podpis úředníka.

V obchodní sféře se tak jeví užitečné použít tyto SMS pro popis klienta. Jako vhodný se jeví následující model:

1. Pokud vzniká např. smlouva mezi dvěmi stranami na internetu, pak vzhledem ke komunikaci klient – server se obě strany nemohou současně podepsat pod jeden dokument. Přirozenější je, že první strana připraví zárodek dokumentu (např. žádost o uzavření smlouvy), podepíše ji SMS podpisem a zašle druhému subjektu.

2. Druhý subjekt ověří podpis prvního subjektu.

3. Druhý subjekt vyhotoví smlouvu s navíc vloženým a SMS podepsaným zárodkem od první strany.

4. Druhý subjekt takto vyhotovenou smlouvu podepíše AdES a vloží časové razítko, tj. vytvoří AdES-T.

5. Ověří se podpis druhého subjektu a výsledná smlouva se uloží do úložiště. To lze provést na obou stranách.

Tím bylo vyhověno liteře Zákona o archivnictví a spisové službě. V čem je problém? Problém je v tom, že jsme neřekli, jak první strana podepíše zárodek smlouvy. Cílem je využít jednorázové heslo zaslané přes SMS. Pokud toto heslo vznikne aplikací jednocené funkce na zárodek, pak se jedná o elektronický podpis. nebo to snad nejsou *„data v elektronické podobě, která jsou připojena k jiným elektronickým datům nebo jsou s nimi logicky spojena a která slouží jako metoda autentizace“*? SMS vzniklá jednocestnou funkcí aplikovanou na dokument je „ten SMS podpis“.

Jednocestná funkce by měla obsahovat nějaké tajemství, aby nebyla známa třetí straně (útočníkovi).

Nelze to použít pro komunikaci se státní moci (vylučuje to § 11 Zákona o elektronickém podpisu), ale v obchodní komunikaci si subjekty mohou např. písemně dohodnout, že SMS podpis budou vzájemně uznávat. Výsledkem je, že:

- Kryptologové budou skřípat zuby, ale je to user friendly.
- Je to elektronický podpis dle zákona, ale není to „uznávaný podpis“. Obchodní smlouvou se ale mohou obě strany dohodnout, že si takový podpis budou vzájemně uznávat.

Soumrak SMS podpisu

SMS podpis je bezesporu kontroverzní. Ránu mu ale zasadila technologie – nové mobilní telefony, které se tváří jako osobní počítač.

Základní výhodou SMS podpisů bylo, že mezi aplikací a uživatelem jsou navázány dva nezávislé komunikační kanály (jeden na počítač a druhý do mobilu). Pro útočníka je obtížené napadnou oba kanály najednou. Jenže dnešní uživatelé používají často „chytrý mobil“ na kterém běží jak aplikace, tak i přijámání SMS. Takže pro útočníka příležitost.

Dynamické biometrické podpisy

Vlastnoruční podpis je ve své podstatě biometrickým údajem. Myšlenka vložit obrázek vlastnoručního podpisu do elektronického dokumentu je vcelku běžná. Takový podpis se běžně neuznává jako podpis zejména z následujících důvodů:

1. Není pevně připojen k dokumentu (obrázek je možné vyměnit).

2. Autentizace osoby spojené s podpisem je sporná, protože obrázek se snadno zkopíruje nebo i zfalšuje.

Dynamický, rukou psaný, biometrický, elektronický podpis se neskenuje jen jako obrázek výsledného podpisu, ale podepisující se osoba se podepisuje na tabletu, který (kromě obrázku podpisu) navíc snímá dynamiku podpisu. Zfalšování takového podpisu je téměř nemožné.

Postup je tedy takový, že podepisovaná osoba vytvoří dynamický podpis, který vloží do dokumentu. Dokument se i s tímto podpisem uzamkne např. PKI podpisem čímž se dokument pevně sváže s podpisem. Model je vlastně obdobný SMS podpisu.

Na závěr znovu odcitujeme Směrnici 1999/93/EC Evropského parlamentu a rady o elektronickém podpisu: *„ ‚elektronickým podpisem‘ se rozumí data v elektronické podobě, která jsou připojena k jiným elektronickým datům nebo jsou s nimi logicky spojena a která slouží jako metoda autentizace“*.

A může začít diskuse jestli dynamický bioemtrický podpis tuto definici splňuje nebo ne. Byť jsem se více jak 10 let zabýval elektronickým podpisem na bázi PKI, tak můj názor je, že splňuje.

Závěr

Extrémní lpění na kryptografické perfektnosti elektronického podpisu (krátká platnost certifikátů a časových razítek, komplikace s vydáváním certifikátu, vysoká cena atd.) vedou k odklonu veřejnosti od stávajících forem elektronického podpisu. Dokonce dne 24. 8. 2011 na serveru http://ihned.cz se v článku „Vláda kývla na reformu daní“ uvádí, že vláda si bere jako svůj cíl:

- Rozšíření uplatnění daňové informační schránky, která bude umožňovat jednoduchou komunikaci se správcem daně, včetně možnosti elektronických podání, která nebudou muset být opatřena elektronickým podpisem.

Zatímco před 10 lety bylo zavedení elektronického podpisu na bázi PKI chápáno jako novátorství, tak dnes se za přínos bere jeho odstranění.

Literatura

[1] ETSI: CMS Advanced Electronic Signatures, ETSI TS 101 733.

[2] IETF RFC-2315: PKCS #7: Cryptographic Message Syntax Version 1.5.

[3] IETF RFC-5652: Cryptographic Message Syntax (CMS).

[4] ETSI: PDF Advanced Electronic Signature Profiles, ETSI TS 102 778-1. až 6.

[5] ETSI: XML Advanced Electronic Signatures, ETSI TS 101 903.

[6] W3C: XML Signature Syntax and Processing.

[7] SMĚRNICE EVROPSKÉHO PARLAMENTU A RADY 1999/93/ES ze dne 13. prosince 1999 o zásadách Společenství pro elektronické podpisy.

Sledování rozsáhlé počítačové infrastruktury

Pavel Tuček

E-mail: tucek@ics.muni.cz

Abstrakt

S rostoucím počtem počítačů v síti rostou i možnosti a rozsah útoků. Ať už se jedná o útok v rámci lokální sítě nebo o útok do sítě cizí, důsledky jsou vždy nepříjemné. Zde je třeba si uvědomit, že sledování počítačů není spásné samo o sobě. Abychom získali komplexní informaci o stavu sítě, či jinak řečeno infrastruktury, je třeba sledovat i související prvky jako jsou směrovače, síťové přepínače, servery a služby, které s chodem celé infrastruktury souvisí. Všechny zaznamenané události je potřeba analyzovat a dát do správných souvislostí. Teprve tehdy získáme informaci o stavu infrastruktury. Tento příspěvek pojednává o motivaci pro vývoj nového monitorovacího systému, jeho komponentách a vytyčených cílech. Vývoj probíhá v rámci na Ústavu výpočetní techniky MU v rámci Oddělení vývoje systémových služeb, které má aktuálně ve správě více než 1 200 počítačů a 50 serverů.

1 Proč sledovat infrastrukturu?

Stejně jako každý rozumný člověk chodí na preventivní prohlídky ke svému lékaři a zubaři, tak by rozumná počítačová infrastruktura měla chodit za svým „lékařem“, tedy za svým správcem. Leč počítačová infrastruktura nemá nožičky, takže v tomto případě by to mělo fungovat opačně – pečlivý správce by měl pravidelně kontrolovat jím spravovanou počítačovou infrastrukturu. Pokud tak neučiní, jistou dobu bude infrastruktura pravděpodobně fungovat, ale postupem času bude docházet k problémům a výpadkům. Správce bude tyto problémy řešit až po nahlášení uživateli, tedy typicky pod tlakem a pouze horkou jehlou. Dlouhodobě neudržitelné řešení.

Předpokládejme, že správce není úplně neznalý a logy na počítačích, síťových prvcích, mail serveru a dalších zdrojích prochází. Bude tedy mít určitý přehled o své infrastruktuře, ale zisk tohoto přehledu ho stojí takřka veškerý pracovní

čas. Uživatel tedy opět nebude spokojen, protože není kdo by mu poskytl podporu. Dalším evolučním krokem tedy bude kolekce logů na jednom místě a jejich filtrování. Kontrola výsledku tohoto procesu už nezabere více než 20 minut denně a uživatel bude konečně spokojený. Tedy pokud uvažujeme menší podnik s jedním dvěma servery, maximálně pár desítkami počítačů a jedním síťovým prvkem.

Co ale dělat ve chvíli, kdy spravujete tisíce počítačových stanic, desítky serverů, Active Directory s 50 tisíci uživatelskými účty, přepínače (switch), směrovače (router), firewally, DNS servery, DHCP servery, webové servery a další služby? Zde je potřeba zcela změnit přístup, protože události už nejsou v řádu desítek, ale v řádu desítek až stovek tisíc. Jak se s takovým množstvím informací vypořádat, si povíme v následující kapitole.

2 Řešení pro tisíce (událostí)

Oblast, která se věnuje správě a sledování infrastruktury jako celku, bývá označována termíny SEM (Security Event Management), SIM (Security Information Management) nebo dnes nejčastěji SIEM (Security Information and Event Management). Přestože jsou tyto termíny běžně zaměňovány, jsou mezi nimi jisté rozdíly. Dle [1] jsou jednotlivé pojmy definovány následovně:

> *SEM – Oblast správy zabezpečení, která se zabývá sledováním infrastruktury, korelací událostí a možnostmi upozornění v reálném čase.*
>
> *SIM – Oblast správy zabezpečení, která se zabývá dlouhodobým ukládáním událostí, jejich analýzou a hlášením případných problémů.*
>
> *SIEM – Oblast správy zabezpečení, která v sobě spojuje prvky SEM a SIM, tzn. zajišťuje sběr bezpečnostních logů generovaných hardwarovou a softwarovou částí infrastruktury a jejich uložení do databáze. Dále zajišťuje analýzu a prezentaci těchto dat, mimo jiné pro přehled o plnění kvality služeb.*

Z definic je vidět, že hlavním rozdílem mezi nástroji SEM a SIM, je orientace na současný stav, versus dlouhodobé uchování informací a vytváření statistik a přehledů. Nástroje SIEM se snaží skloubit funkčnost a výhody předchozích jmenovaných. Vzhledem k tomu, že naší motivací je jak informace o stavu infrastruktury v reálném čase, tak možnost vytvářet dlouhodobé statistiky, tak se budeme nadále věnovat především nástrojům SIEM.

Nástroje SIEM jsou tvořeny komponentami, které zajišťují tyto čtyři oblasti souvisejících se zpracováním dat [2, 3]:

1. *Agregace dat – Sběr dat z mnoha zdrojů, mezi které patří přepínače, firewally, IDS/IPS, netflow sondy, servery, počítačové stanice, databáze, aplikace, atd. V ideálním případě by se mělo jednat o data ze všech prvků, které jsou součástí infrastruktury. Součástí agregace d.*

2. *Korelace událostí – Korelace hledá společné atributy a spojuje události do smysluplných množin. Schopnost dívat se na všechny události v určitém časovém okně nebo sledovat činnost konkrétního uživatele, poskytuje jedinečnou možnost pro zkoumání/vyšetřování bezpečnostních incidentů. Korelace zahrnuje také převedení dat na stejný formát, změnu logické struktury dat pro snazší vyhledávání a jejich uložení dat do databáze.*

3. *Varování – Tato část zajišťuje automatizovanou analýzu dříve připravených množin událostí a generuje varování v případě nalezení problému. Varování jsou typicky rozlišena podle vážnosti problému a podle informačního kanálu, který je použit pro doručení varování zodpovědné osobě.*

4. *Přehledové sestavy – Nástroje SIEM vytváří z dat událostí informační tabulky a grafy, které usnadňují hledání vzorů útoků a identifikaci podezřelých aktivit. Přehledové sestavy současně slouží jako podklady pro zprávy o dodržování kvality a dostupnosti služeb.*

V některých případech bývá korelace událostí rozdělena na dvě oblasti a to konsolidace dat a korelace událostí. Důležité komponenty SIEM řešení už jsme si představili, takže už chybí pouze představení cílů, které by tyto nástroje měly splňovat. Jedná se především o:

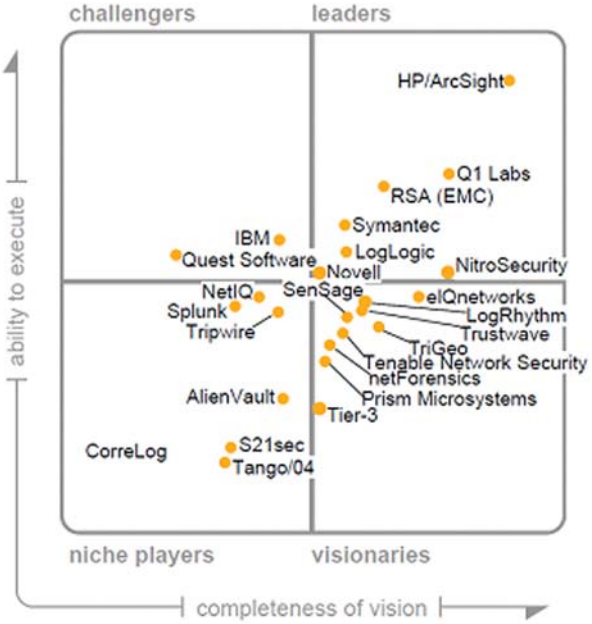
- rychlejší reakce na útoky
- větší úspěšnost detekce útoků
- vyšší efektivita správy infrastruktury a
- automatické vytváření statistik dostupnosti infrastruktury.

I přesto, že se jedná o poměrně mladou oblast na poli IT (první produkty typu SIEM se začaly objevovat před cca 12 lety [4]), SIEM řešení nabízí většina velkých hráčů na poli bezpečnosti IT. Mezi nejvýznamnější řešení na trhu patří:

- Enterprise Threat and Risk Management (ETRM) od společnosti ArcSight (dceřiná společnost HP), [5]
- SecureVue od společnosti eIQnetworks, [6]
- Sentinel od společnosti Novell (po akvizici E-Security), [7]

- Trustwave SIEM od společnosti Trustwave (po akvizici Intellitactics), [8]
- enVision od společnosti RSA/EMC^2 (po akvizici Network Intelligence), [9]
- NitroView Enterprise Log Manager od společnosti NitroSecurity (po akvizici LogMatrix), [10]
- Advanced SIEM and Log Management od společnosti Sensage, [11]
- Symantec Security Information Manager (SSIM) od společnosti Symantec [12] a
- SolarWinds Log & Event Manager od společnosti TriGeo/SolarWinds. [13]

Všechna uvedená řešení se umístila v kvadrantu lídrů či vizionářů v „Magickém kvadrantu pro SIEM“ v roce 2011 [14], průzkumu prováděném každoročně společností Gartner



Obr. 1 „Magický kvadrant pro SIEM“ v roce 2011, Gartner [14]

3 Motivace pro vývoj vlastního řešení

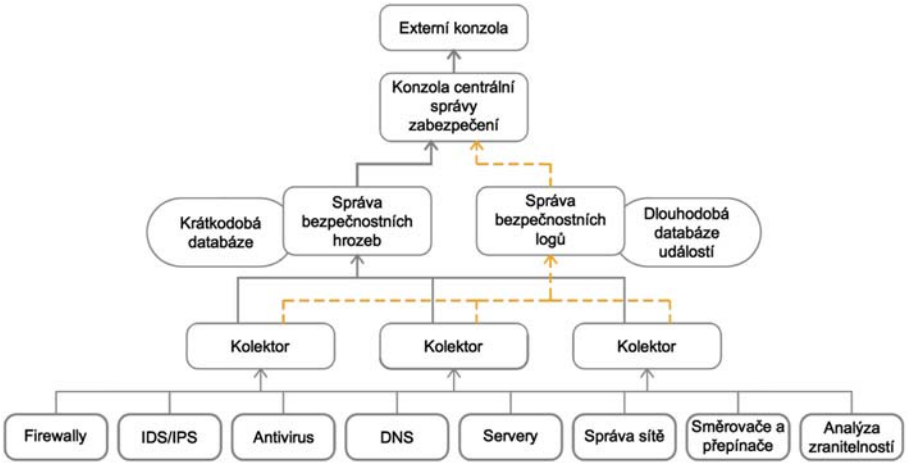
Vývoj vlastního řešení je jednou z možností, jak se postavit k problému, který právě řešíte Zda se jedná o řešení dobré či špatné se ukáže až v delším časovém horizontu a v obou případech to stojí nemalé množství prostředků. V našem případě jsme pro rozhodování vyhodnotili jako kritické faktory: cena řešení, problematika konfigurace a nasazení, podpora, výkon a přínos. Každý z těchto faktorů si nyní blíže projdeme.

Podle [15] začíná cena za nasazení SIEM nástrojů ve velkém podniku (enterprise bussiness) na $ 75 000, ale může vystoupat až k $ 500 000 (u nadnárodních korporací ještě výš). I počáteční suma je při současném kurzu dolaru ke koruně více jak 1 300 000 Kč není zanedbatelná suma peněz. Konfigurace a nasazení SIEM řešení také nebývá procházkou růžovým sadem. Dle [16] není neobvyklé, že při nasazení ve větších podnicích dochází ke skluzu 6–8 měsíců oproti očekávání a konečná cena nasazení se navýší o $ 250 000–$ 500 000. Dalším rizikem je podpora dodaného řešení, bez které je řešení dlouhodobě nepoužitelné. Zde se jedná především o vlastní špatné zkušenosti, které jsme měli při nasazování nástrojů určených do větších podniků. Posledním z měřitelných faktorů je výkon. Přestože každý producent SIEM řešení prezentuje výkon svého řešení, tak uváděné hodnoty mají jen malou vypovídající hodnotu. Vrchní mez výkonu se pohybuje kolem 10 tisíc vyhodnocených událostí za vteřinu (dále EPS, Events Per Second), což není špatné číslo. Ale je dobré se nad touto hodnotou zamyslet, vždyť jak říká klasik: „Nevěřím žádné statistice, kterou jsem sám nezfalšoval.“ Testy probíhají v laboratorním prostředí, prvky zapojené v řešení jsou výrobci dobře známy a jsou nakonfigurovány pro nejvyšší výkon. Lze se oprávněně domnívat, že realita tomuto nebude odpovídat a je jen na našem pesimismu hádat, o kolik maximální hodnota EPS poklesne. Vzhledem k tomu, že cílovým subjektem je Masarykova univerzita, čítající tisíce počítačů, stovky síťových prvků, stovky serverů, tisíce zaměstnanců a desetitisíce studentů... není těžké si představit, že v krizovém momentu útoku je počet událostí nutných ke zpracování výrazně vyšší než 10 tisíc EPS. Pokud je nezpracujeme, tak přijdeme o sledování v reálném čase a v horším případě se nám je nepodaří ani uložit pro zpětnou analýzu.

Posledním faktorem, který ovlivnil naše rozhodování, byla zajímavost a komplexnost SIEM řešení. Jedná se o oblast, ve které je třeba řešit mnoho netriviálních otázek jako např. anonymita, komplexní zpracování událostí (Complex Event Processing, dále CEP), síťová bezpečnost, bezpečnost operačních systémů (dále OS), dolování dat (datamining), protokoly pro přenos zpráv a mnohé další. Vzhledem k tomu, že Masarykova univerzita je mimo vzdělávací instituci především institucí výzkumnou, je řešení těchto otázek jejím denním chlebem a část těchto otázek mohou zpracovat doktorští studenti v rámci své dizertace. Zkusíme tedy spojit příjemné s užitečným.

4 Architektura

Stejně jako pro každé jiné řešení, i pro řešení SIEM je důležitý návrh architektury. Optimální architektura SIEM obsahuje všechny komponenty, které je vidět na obrázku 2.



Obr. 2 Optimální architektura řešení SIEM [15]

Každá komponenta má svá specifika a důvod pro existenci. Spodní řadu v obrázku přeskočíme, protože ta představuje pouze různá zařízení a služby, z kterých máme zájem sbírat logy. Onen sběr logů ale není tak přímočarý, jak by jeden čekal. Některá zařízení či služby logují do souborů, kdežto jiná odesílají proud dat, některá ukládají v lidsky čitelném kódu (text, XML), jiná v binárce, některá podporují šifrování ukládaných logů, jiná ne. Je zřejmé, že takto to nepůjde. Do sběru logů zapojíme tzv. kolektory, které mají za úkol sběr logů a odfiltrování odlišností na straně zařízení a služeb. Z kolektorů data putují souběžně do dvou míst: do centra pro správu bezpečnostních hrozeb a do centra pro správu bezpečnostních logů. Centrum pro správu bezpečnostních hrozeb je ta část řešení SIEM, která zajišťuje zpracování logů v reálném čase (zpracování, korelace, uložení do databáze). Data ukládá do databáze optimalizované na rychlý přístup a jejich stáří by nemělo být větší než 30 dní. Centrum pro správu bezpečnostních logů má za úkol ukládání dat do dlouhodobé databáze událostí. Ta by měla být optimalizována pro kompresi a indexování dat a kapacitně by měla pokrýt alespoň 18 měsíců.

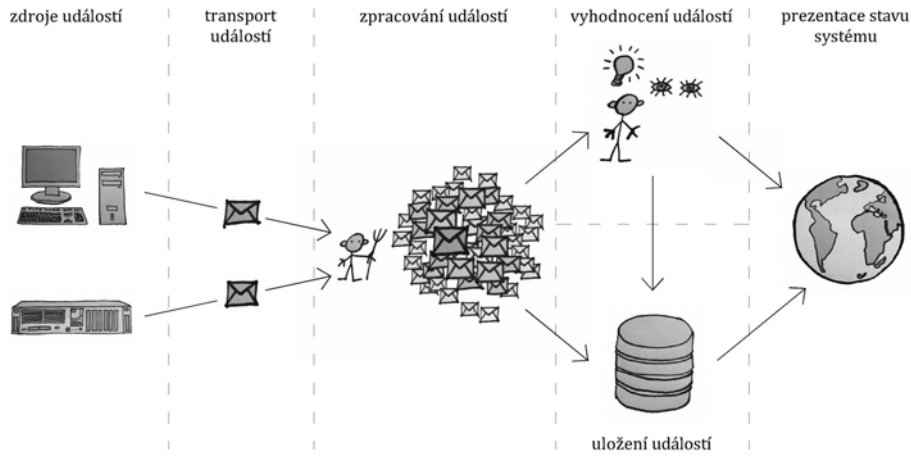
Konzola centrální správy zabezpečení slouží pro prezentaci současného stavu infrastruktury bezpečnostnímu personálu a manažerům. Jedná se o rozhraní pro

každodenní operace, které by mělo mimo aktuální stav infrastruktury, nabídnout také přístup k historickým záznamům. Externí konzola poskytuje možnost připojení mimo podnikovou síť s omezeným přístupem k datům.

5 Sledování Microsoft Active Directory infrastruktury

Aktuálně máme v rámci Oddělení vývoje systémových služeb (dále OVSS) Ústavu výpočetní techniky (dále ÚVT) MU ve správě více než 1 200 počítačů a 50 serverů s operačními systémy od společnosti Microsoft. Srdcem této infrastruktury je Active Directory doména UCN (University Computer Network), která zastřešuje univerzitní počítačovou síť tvořenou převážně studentskými, ale nyní už i zaměstnaneckými počítači. Sledování této infrastruktury a souvisejících služeb bylo primární motivací pro nasazení řešení SIEM. Proč jsme se rozhodli pro vlastní řešení bylo zodpovězeno výše, takže se rovnou podíváme na současný stav.

Oproti optimální architektuře se naše stávající řešení liší. Vrstva kolektorů je u nás popsána „zdroji událostí“ a „transportem událostí“. Centrum pro správu bezpečnostních logů je tvořeno nástrojem CloverETL a cca 20 tabulkami v databázi Oracle 10g Enterprise Edition, kterou spravuje Oddělení správy informačních systémů ÚVT MU. Centrum pro správu bezpečnostních hrozeb s krátkodobou databází ani konzoly centrální správy zabezpečení zatím nemáme realizovány. Původně plánovaná architektura sledování je vidět na obrázku 3 a následovat bude podrobnější popis dvou agentů pro kolekci logů.



Obr. 3 Původně plánovaná architektura

5.1 WinMon

WinMon je název služby, která má za úkol sledování zabezpečení stanic a serverů s OS Windows a generování dílčích zpráv o jejich stavu. Aktuálně sledované oblasti jsou:

- antivirový systém,
- firewall,
- síťová rozhraní,
- updaty a stav jejich instalace,
- uživatelé a skupiny,
- základní parametry operačního systému (uptime, datum instalace, verzi OS, ...) a
- logy operačního systému (Eventlogs).

Při získávání informací bylo dbáno na to, aby způsob jejich získávání byl pokud možno univerzální pro všechny operační systémy Windows a všechny řešení třetích stran (kromě firewallu – doteď totiž nebyla motivace sledovat FW jiných výrobců). V praxi to tedy znamená, že informace o uživatelích a skupinách, síťových rozhraních a základních parametrech OS jsou sledovány pomocí prostředků dostupných ve všech OS Windows od Windows XP SP3 dále (starší systémy jsme neuvažovali):

- WMI,
- registry a
- patřičná API v .NETu

U antivirových systémů je jejich stav zjišťován z Microsoft Security Center. Sledování firewallu je zatím implementováno pro řešení dodané přímo od výrobce OS, tedy Windows Firewall.

Generování zpráv o stavu systému probíhá dle následujících scénářů:

- Při prvním spuštění služby WinMon dojde k analýze stavu počítače/serveru a vygenerování relevantní zprávy.
- Část oblastí (typicky těch, které mají stav uložený v registrech) je monitorována na základě zachytávání systémových událostí pomocí služby Platform Invocation Services [17] (známé též jako PInvoke). Ve chvíli zachycení systémové události týkající se sledované části registru, dojde k vytvoření zprávy o změně.

- Ostatní oblasti jsou kontrolovány v pravidelných intervalech, které je možné definovat dle potřeby, a v případě zjištění rozdílu nového stavu od stavu předchozího (přidání, odebrání či změna v některém sledovaném parametru), dojde k vytvoření patřičné zprávy.

- Posledním typem generovaných zpráv je kompletní zpráva o stavu systému (obdoba toho, co se děje při prvním spuštění služby) s tím, že interval generování této zprávy je možné nastavit dle potřeby (typicky jednou za 24 hodin).

Služba je instalována pomocí MSI balíku, což je ideální pro distribuci pomocí skupinové politiky Active Directory. V rámci této politiky je také možno službu konfigurovat.

5.2 Namtar

Namtar je balíček tří služeb, které se jmenují WinLogonEventTrace, WLETDispatcher a WLETSpooler.

WinLogonEventTrace má na starosti sledování a zpracování událostí typu logon, logoff, startup, shutdown, lock a unlock na hostitelských operačních systémech. Pro OS Windows XP je aplikace WinLogonEventTrace implementována jako tzv. notifikační balíček služby Winlogon. Jedná se vlastně o knihovnu DLL, která přijímá zprávy od služby Winlogon v okamžiku, kdy vznikne některá z výše uvedených událostí. Po přijetí zprávy je tato událost zpracována, což v praxi znamená získání požadovaných informací o události (typ události, čas vzniku události, uživatel, který událost způsobil, stanice, na které k události došlo, atd.). Získané události jsou zaznamenány následnými kroky:

- zápisem do EventLogu,

- zápisem do textového souboru WinLogonEventTrace a

- vygenerováním XML souboru, který je uložen na disku.

Od OS Windows Vista však technologie notifikačních balíčků pro službu Winlogon není podporována a proto se v současné době pracuje na implementaci nové technologie. Tou je příjem upozornění (notifikací) od služby System Event Notification Service (SENS) [18]. Je to jeden z přístupů, který je možné použít jako náhradu za notifikační balíčky služby Winlogon. Tato technologie byla zvolena především z toho důvodu, že poskytuje upozornění na největší počet událostí původní služby Winlogon. Prakticky se jedná o všechny typy sledovaných událostí, kromě událostí startup a shutdown. Z důvodu absence těchto dvou událostí, byla aplikace přijímající upozornění od služby SENS implementována jako služba spustitelná při startu počítače – je možné detekovat zapnutí (startup) a vypnutí

(shutdown) počítače. Tato služba byla pojmenována WLETCatcher a slouží pro detekci požadovaných událostí. Při implementaci byla použita technologie Component Object Model (COM). Pro zpracování události je použitá mírně upravená aplikace WinLogonEventTrace, která už není napojená na službu Winlogon, ale přímo na službu WLETCatcher. WinLogonEventTrace zabezpečí samotné zpracování detekované události.

Jak již bylo zmíněno výše, součástí balíčku Namtar jsou služby WLETDispatcher a WLETSpooler. Na rozdíl od aplikace WinLogonEventTrace, která slouží ke sledování a generování zpráv, tyto slouží především jako infrastruktura pro přenos vzniklých zpráv z jednotlivých pracovních stanic na server, a odtud případně na další servery, pokud je cesta delší než jeden „skok“. Služba WLETDispatcher přitom běží na straně, která záznam odesílá a služba WLETSpooler na straně, která záznam přijímá.

Samotný přenos je založený na technologii Remote Procedure Call (RPC), což je vzdálené volání procedury. Služba WLETDispatcher běží na hostitelské pracovní stanici a čeká na vznik XML souboru ve sledovaném adresáři. Na základě detekce nového souboru, naváže spojení se službou WLETSpooler a přepošle jí obsah daného souboru. Po úspěšném přenosu služba WLETDispatcher tento XML soubor z daného adresáře vymaže a čeká na vznik dalších souborů. Služba WLETSpooler běží na serveru a čeká na volání služby WLETDispatcher z některé pracovní stanice. Po vytvoření spojení a přijetí obsahu přenášeného záznamu, vytvoří XML soubor ve vybraném adresáři a vloží do něj přijatý záznam. Následně čeká na další volání služby WLETDispatcher.

Balíček Namtar se instaluje, obdobně jako WinMon, pomocí MSI balíku a stejně tak je v rámci skupinové politiky možné konfigurovat nastavení jednotlivých částí balíku.

5.3 CloverETL

Pro zpracování dat je aktuálně využívána služba původně naprogramována pro balík softwaru Namtar, ale ta bude v blízké době kompletně nahrazena platformou CloverETL [19], která zajišťuje zpracování dat z různých informačních zdrojů, jejich transformaci na data se stejnou strukturou a následné poskytnutí dat dalším službám či nástrojům.

CloverETL je produktem firmy Javlin a jedná se o zástupce softwaru označovaného zkratkou ETL (Extract, Transform, Load). ETL je typický proces využívaný v databázích a datových skladech, kdy:

- extrahujeme data z vnějších zdrojů,
- transformujeme je tak, aby odpovídala našim potřebám,
- následně je vložíme do databáze nebo jako vstup dalším aplikacím.

CloverETL je nabízen v několika edicích, přičemž engine je nabízen také jako open source pod licencí LGPL.

6 Další vývoj

V další práci nás čeká přidání centra pro správu bezpečnostních hrozeb, které zajistí vyhodnocení stavu systému v reálném nebo téměř reálném čase. Zpracování velkého množství událostí v reálném čase je pravděpodobně nejdůležitější komponentou tohoto systému. Typické úlohy, které potřebujeme řešit, jsou:

- přiřazení významu jednotlivým událostem,
- vyhodnocení událostí a rozhodnutí, zda se jedná o dobrý stav (v našem případě bezpečný) či ne,
- generování odvozených událostí,
- rozhodování o adekvátních reakcích,
- diagnostika problémů a jejich příčin,
- „předvídání“ událostí.

Z posouzených technologií se jako ideální ukazuje technologie Complex Event Processing (CEP), která všechny uvedené scénáře řeší. Základními principy technologie CEP jsou:

- detekce vzorů událostí,
- odvozování komplexních (složených) událostí z událostí jednoduchých (základních),
- modelování hierarchií událostí,
- sledování vazeb mezi jednotlivými událostmi, přičemž je věnován velký důraz na časovou složku.

Výhodou technologie CEP je také dostupnost programovacích jazyků a softwarových nástrojů pro integraci této technologie do větších systémů. V posledních letech se objevilo několik implementací technologie CEP, které pokrývají různé přístupy k řešení základních principů CEPu. Jedná se jak o placené implementace velkých korporací jako IBM, tak o řešení volně dostupná jako open source. Jednotlivá řešení se liší především v jazyce pro formulaci dotazů (proudový nebo na základě pravidel), ve výkonu, dostupnosti podpůrných nástrojů (GUI pro návrh dotazů, atd.) a v neposlední řadě v ceně.

Kandidát, který byl vybrán pro testovací implementaci je Microsoft StreamInsight [20] Poskytuje výhodu především v návaznosti na ostatní technologie společnosti Microsoft, se kterými má OVSS ÚVT MU dlouholeté zkušenosti. Jako dotazovací jazyk se využívá Microsoft LINQ a princip práce je založený na reaktivním programování. Co se architektury týká, StreamInsight je implementován jako samostatná služba systému Windows, která poskytuje koncové body pro vstupní a výstupní události, nebo je dostupný ve formě knihovny DLL, kterou je možné zakomponovat do vlastního řešení.

Poslední součástí monitorovacího systému je konzola centrální správy zabezpečení, která má umožnit vizualizaci současného stavu infrastruktury. Kromě zobrazení aktuálního stavu celé infrastruktury bude možné zobrazit jednotlivé části infrastruktury až na úroveň konkrétního zdroje logů, kde bude vidět stav sledovaných komponent a historické záznamy

V neposlední řadě jsme označili několik dalších komponent, které bychom do systému rádi zapojili. Jedná se především o komponenty, které budou dodávat další vstupní data:

- sledování linuxových operačních systémů,
- sledování síťových prvků (Cisco, HP, 3COM, ...),
- sledování zabezpečovacích systémů, které OVSS ÚVT MU vyvíjí,
- sledování zálohování stanic a serverů (integrace zálohovacího řešení Bacula [21]),
- sledování autentizačních bran využívajících doménu UCN (Radius, Shibboleth [22] a jiné)

7 Závěr

V předchozích odstavcích jsme se zamysleli nad vhodností, či spíše nutností sledování infrastruktury. Nadefinovali jsme pojmy SEM, SIM a SIEM. Přiblížili jsme si cíle očekávané při nasazování SIEM a čtyři oblasti zpracování dat v řešeních SIEM. Uvedli jsme si příklady produktů SIEM, které patří mezi nejvýznamnější na trhu. Zamysleli jsme se nad motivací vyvíjet vlastní SIEM řešení. Od páté kapitoly dál jsme si představili vznikající řešení pro sledování infrastruktury v reálném nebo téměř reálném čase, komponenty hotové i komponenty, které musí být teprve dokončeny.

Pevně doufám, že za rok se budeme moci pochlubit dalšími posuny v práci na plnohodnotném monitorovacím nástroji, který budeme moci zařadit do kategorie SIEM.

Poděkování

Tento článek by pravděpodobně nevznikl nebýt správného nasměrování Vaškem Lorencem, který se touto problematikou aktuálně zabývá ve společnosti Honeywell. Dále tímto děkuji svým kolegům Víťovi Bukačovi a Andrey Číkové za podněty a pomoc V neposlední řadě děkuji svému školiteli, Vashkovi Matyášovi, za motivaci pokračovat dál.

Literatura

[1] Wikipedia: *Security information and event management*, Wikipedia, the free encyclopedia, 7 červenec 2011. [Online]. Available: http://en.wikipedia.org/wiki/Security_information_and_event_management. [Přístup získán 5 září 2011].

[2] A. Lane: *Understanding and Selecting SIEM/LM: Use Cases, Part 1*, Securosis, 30 duben 2010. [Online]. Available: http://securosis.com/blog/understanding-and-selecting-siem-lm-use-cases-part-1. [Přístup získán 5 září 2011].

[3] A. Lane: *Understanding and Selecting SIEM/LM: Use Cases, Part 2*, Securosis, 3 květen 2010. [Online]. Available: http://securosis.com/blog/understanding-and-selecting-siem-lm-use-cases-part-2. [Přístup získán 5 září 2011].

[4] Wikipedia: *Security event manager*, Wikipedia, the free encyclopedia, 25 srpen 2011. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Security_event_manager. [Přístup získán 5 září 2011].

[5] ArcSight: *The ArcSight ETRM Platform*, ArcSight, 3 srpen 2011. [Online]. Available: http://www.arcsight.com/products/. [Accessed 5 září 2011].

[6] eIQnetworks: *eIQnetworks – Unified Situational Awareness – SecureVue*, eIQnetworks, [Online]. Available: http://www.eiqnetworks.com/securevue/securevue.php. [Accessed 5 září 2011].

[7] Novell: *Sentinel*, Novell, 2 květen 2011. [Online]. Available: http://www.novell.com/products/sentinel/. [Accessed 5 září 2011].

[8] Trustwave: *Trustwave SIEM*, Trustwave, [Online]. Available: https://www.trustwave.com/siem/siem.php. [Accessed 5 září 2011].

[9] RSA/EMC: *enVision*, RSA/EMC, [Online]. Available: http://www.rsa.com/node.aspx?id=3170. [Accessed 5 září 2011].

[10] NitroSecurity: *NitroView Enterprise Log Manager*, NitroSecurity, [Online]. Available: http://www.nitrosecurity.com/solutions/log-management/. [Accessed 5 září 2011].

[11] Sensage: *Advanced SIEM and Log Management*, Sensage, [Online]. Available: http://www.sensage.com/content/advanced-siem-and-log-management. [Accessed 5 září 2011].

[12] Symantec: *Security Information Manager*, Symantec, [Online]. Available: http://www.symantec.com/business/products/multimedia.jsp?pcid=pcat_info_risk_comp&pvid=929_1. [Accessed 5 září 2011].

[13] TriGeo/SolarWinds: *SIEM Network Security Solution*, TriGeo/SolarWinds, [Online]. Available: http://www.trigeo.com/products/. [Accessed 5 září 2011].

[14] M. Nicolett, K. M. Kavanagh: *Magic Quadrant for Security Information and Event Management*, Gartner, 12 zvěten 2011. [Online]. Available: http://www.arcsight.com/collateral/whitepapers/Gartner_Magic_Quadrant_2011.pdf. [Accessed 5 září 2011].

[15] D. Swift, *A Practical Application of SIM/SEM/SIEM Automating Threat Identification*, 23 prosinec 2006. [Online]. Available: http://www.sans.org/reading_room/whitepapers/logging/a_practical_application_of_sim/sem/siem_automating_threat_identification_1781. [Přístup získán 5 září 2011].

[16] A. Williams: *The Future of SIEM – The market will begin to diverge*, Amrit Williams Blog, 1 leden 2007. [Online]. Available: http://techbuddha.wordpress.com/2007/01/01/the-future-of-siem-the-market-will-begin-to-diverge/. [Přístup získán 5 září 2011].

[17] Microsoft: *Winlogon Notification Packages*, Microsoft, 14 červenec 2011. [Online]. Available: http://msdn.microsoft.com/en-us/library/aa380545(VS.85).aspx. [Accessed 5 září 2011].

[18] Microsoft: http://msdn.microsoft.com/en-us/library/aa376860(VS.85).aspx, Microsoft, 3 červen 2010. [Online]. Available: http://msdn.microsoft.com/en-us/library/aa376860(VS.85).aspx. [Accessed 5 září 2011].

[19] Javlin: *CloverETL*, Javlin, [Online]. Available: http://www.cloveretl.com/. [Accessed 5 září 2011].

[20] Microsoft: *Microsoft StreamInsight – Complex Event Processing*, Microsoft, [Online]. Available: http://www.microsoft.com/sqlserver/en/us/solutions-technologies/mission-critical-operations/complex-event-processing.aspx#. [Accessed 5 září 2011].

[21] Bacula.org: *Bacula – The Open Source Network Backup Solution*, Bacula.org, [Online]. Available: http://www.bacula.org/en/. [Accessed 5 září 2011].

[22] Internet2: *Shibboleth*, Internet2, 25 červenec 2011. [Online]. Available: http://shibboleth.internet2.edu/. [Accessed 5 září 2011].

[23] CRN: *SIEM: A Market Snapshot*, CRN, 5 únor 2007. [Online]. Available: http://www.crn.com/news/security/197002909/siem-a-market-snapshot.htm. [Přístup získán 5 září 2011].

[24] Wikipedia: *Security information management*, Wikipedia, the free encyclopedia, 5 červenec 2011. [Online]. Available: http://en.wikipedia.org/wiki/Security_information_management. [Přístup získán 5 září 2011].

Uzamčená firemní síť

Ondřej Ševeček

E-mail: ondrej@sevecek.com

Abstrakt

Podnikovým sítím hrozí mnoho nebezpečí od vnějších útočníků a všichni jsou si těchto hrozeb jako jsou viry, spam a hackerské útoky, velmi dobře vědomi. To, co si uvědomuje jen málo firemních prostředí je hrozba zevnitř. Mnohem větší riziko představují zaměstnanci, nebo externisté, dodavatelé, brigádníci, nebo různí servisní pracovníci a údržbáři. Mají fyzický přístup k počítačovému a síťovému vybavení a mohou mnohem snáze získat přístup k datům společnosti. Přednáška se snaží přiblížit nejpalčivější problémy stávajících sítí na platformě Microsoft a nabídnout jejich řešení za pomoci bezpečnostních technologií, které jsou od výrobce k dispozici.

Zabezpečení bezdrátových sítí založených na protokolu IEEE 802.11 v aplikaci rozsáhlých distribučních sítí

Martin Zadina, Jan Nagy, Petr Hanáček

E-mail: izadina@fit.vutbr.cz, inagy@fit.vutbr.cz, hanacek@fit.vutbr.cz

Abstrakt

Různé formy bezdrátové komunikace jsou stále oblíbenější. Rádiové vlny použité jako komunikační médium se šíří od vysílače v prostoru s klesající intenzitou (v závislosti na vzdálenosti a překážkách) a mohou být zachyceny i mimo uvažovanou oblast pokrytí. (Tato ale může být dostatečná pro plnohodnotný příjem signálu.) Z tohoto důvodu by neměla být opomíjena otázka zabezpečení bezdrátové komunikace. V naší práci se zaměříme na několik existujících technologií určených pro bezdrátové lokální sítě, které mohou tvořit prvky sítí většího rozsahu (MAN, WAN). Zmíníme standardizované (IEEE 802.11, 802.16, HiperLAN/MAN), plně proprietární (Motorola Canopy) i kombinované technologie (Ubiquity Networks AirMAX, Mikrotik NStreme, Nv2) určené pro tento účel, jejich bezpečnostní mechanizmy a problematiku nasazení bezpečnostních protokolů sítí 802.11 v aplikaci sítí pevného bezdrátového přístupu.

Klíčová slova: bezdrátové sítě, IEEE 802.11, FWA (Fixed Wireless Access), bezpečnost

1 Úvod

Bezdrátové sítě jsou čím dál oblíbenější prostředek pro přenos dat. Jelikož je k přenosu dat použito rádiových vln, neměla by být opomíjena otázka zabezpečení těchto sítí. Rádiové vlny se šíří s nižší intenzitou signálu i mimo požadovanou oblast pokrytí, a pokud není síť zabezpečena, může se k takové síti připojit kdokoliv v oblasti pokrytí, odposlouchávat či manipulovat s komunikací, využívat

a zneužívat prostředky, které síť poskytuje. Pasivním útokům odposlechem nelze zamezit, pouze omezit zajištěním důvěrnosti dat.

V naší práci se zmíníme o existujících technologiích určených pro bezdrátové lokální sítě, které mohou tvořit prvky distribučních sítí (tj. sítí zprostředkovávajících přenos dat) většího rozsahu (MAN, WAN). Pozornost věnujeme zejména sítím podle protokolu IEEE 802.11, které jsou dnes v drtivé míře nejpoužívanější.

2 Současné technologie bezdrátových lokálních sítí

IEEE 802.11 je množina standardů pro implementaci lokálních bezdrátových sítí (WLAN). Zařízení pracující dle standardu IEEE 802.11 a splňující dané požadavky může nést označení wifi. Technologie wifi je aktuálně běžnou součástí většiny nových přenosných zařízení – laptopy, tablety, čtečky elektronických knih, chytré telefony – a další tržní trendy (řízení spotřeby energie, rozvoj datových, hlasových a video služeb pro digitální domácnost, bezpečnostní aplikace) tuto penetraci podporují.

K dispozici jsou i další standardizované technologie – IEEE 802.16 nebo ETSI HiperLAN/MAN, plně proprietární technologie – Motorola Canopy a technologie založené na proprietárních rozšířeních standardizovaných technologií – Ubiquity AirMAX, Mikrotik NStreme a NV2 (všechny založeny na 802.11).

Z pohledu reálných aplikací vede počtem i oblastmi nasazení standard IEEE 802.11 včetně proprietárních nadstaveb protokolu. Často se lze setkat v reálném nasazení i s technologií Motorola Canopy, ovšem pouze ve specifické oblasti sítí pevného bezdrátového přístupu (FWA). Ve stejné oblasti aplikace se dá narazit i na IEEE 802.16 (Wimax). Ostatní technologie (především ETSI HiperLAN/MAN) se k reálnému nasazení a rozšíření ve větší míře nedostaly.

3 Bezpečnostní mechanismy v technologiích bezdrátových sítí

HiperLAN

HiperLAN je evropský standard chápaný jako obdoba IEEE 802.11. Postupně bylo schváleno několik verzí tohoto standardu. Fyzická vrstva HiperLANu je podobná IEEE 802.11, oba standardy se pak liší ve vrstvě kontrolující přístup k médiu. Po standardizaci 802.11n byla aktivita HiperLANu utlumena, některé principy byly použity ve standardech 802.11n a IEEE 802.16 (WiMax). HiperLAN/2 umožňuje vzájemnou autentizaci AP a stanice pomocí dvou metod [1]. Jedna z nich je použití sdíleného klíče, další možnost autentizace pomocí RSA.

Při využití RSA dovoluje HiperLAN/2 použití pouze tři daných velikosti klíčů (512, 768 a 1 024 bitů). Problémem je chybějící integrita dat během přenosu. I přes počáteční autentizaci obou stran lze do přenosu vkládat nová data nebo měnit přenášená data. Důvěrnost dat lze volitelně zajistit šifrováním pomocí DES nebo 3DES. Použité klíče jsou unikátní pro každou relaci na rozdíl např. od IEEE 802.11i PSK, kde je použit sdílený klíč.

Canopy

Canopy je systém bezdrátové komunikace vyvinutý společností Motorola. Je postaven na proprietárním protokolu, jehož specifikace nebyly zveřejněny [2], takže ani nelze rozumně postavit radio sniffer. Ten by navíc musel obsahovat proprietární Canopy čip, který není běžně k dispozici. Stejně tak MAC protokol pro vytváření, dekompozici a opakovaný přenos paketů nebyl zveřejněn.

Canopy však nespoléhá jen na tzv. Security through obscurity, nabízí i určité bezpečnostní mechanismy. Pro autentizaci je využíván BAM (Bandwith and Authentication Manager). Ten řídí přístup k síti Canopy. Každý AP může nakonfigurován, aby vyžadoval autentizaci koncové stanice pro přístup k síti. Po obdržení žádosti BAM vygeneruje náhodné číslo, které pošle koncové stanici jako výzvu. Koncová stanice vypočítá odpověď pomocí přednastaveného klíče (nebo autorizačního klíče přiděleného operátorem sítě) a pošle zpět. BAM pak v případě shody odpovědi s vlastním výpočtem (za použití stejného náhodného čísla a autentizačního klíče koncové stanice) přidělí přístup.

Správa klíčů v Canopy se stará o distribuci dvou klíčů a náhodného čísla pro autentizaci. Kromě toho se využívá unikátního přednastaveného sériového čísla každé koncové stanice – to se také účastní procesu autentizace. Jeden ze dvou klíčů je autentizační klíč, který nastavuje operátor sítě do SQL databáze a na koncovou stanici (tam jej může nastavit také koncový uživatel). Tento klíč nelze běžně zobrazit v konfiguračním rozhraní koncové stanice. Druhý klíč je session key (počítán z autentizačního klíče, sériového čísla stanice a náhodného čísla). Session key se počítá na koncové stanici i v BAM. Operátor sítě ani uživatel klíč neuvidí v otevřené podobě. Klíč lze použít pro zajištění důvěrnosti dat pomocí DES (56b) nebo AES (128b). Náhodné číslo je generováno BAM a používá se pro autentizaci koncové stanice.

Wimax (IEEE 802.16)

Implementace bezpečnostních prvků ve WIMAXu byly prováděny s ohledem pro nasazení na větší plochy. Po standardizaci IEEE 802.16e pak přibyly bezpečnostní mechanismy pro mobilní služby a většina z bezpečnostních děr předchozí verze 802.16-2004 byla opravena. Jedním z důvodů, proč je bezpečnost WIMAXu považována za kvalitní, je bezpečnostní protokol na spodní části MAC vrstvy poskytující operativně různé bezpečnostní služby [3].

Pro autentizaci koncové stanice se používá certifikát X.509, který je svázán s MAC adresou koncové stanice, nebo metoda EAP. Doporučený šifrovací algoritmus je RSA. Po autentizaci dojde k výměně tří klíčů, šifrovacího pro komunikaci mezi základnovou a koncovou stanicí a dvou autentizačních kódů na bázi hash funkcí. Důvěrnost dat a jednoho z šifrovacích klíčů je zajištěna pomocí 3DES nebo AES (v závislosti na délce zmíněného šifrovacího klíče). V případě nasazení algoritmu DES je potřeba zajistit náhodné generování IV pro každý paket. Integrita dat a některých řídících zpráv je zajištěna pomocí HMAC algoritmu, protože však není chráněna integrita všech řídících zpráv, může to vést k DoS útokům.

S implementací dalších aplikací do standardu 802.16e (VoIP, real-time aplikace) bude třeba vyřešit některé otevřené bezpečnostní problémy týkající se např. rychlé a bezpečné výměny klíčů při hand-overu, aby se nemusela neustále opakovat kompletní autentizační procedura.

3.1 Bezpečnostní mechanismy IEEE 802.11

Standard 802.11 obsahuje několik bezpečnostních protokolů – WEP, WPA, WPA2. Protokol WPA2 je finálním výstupem standardu 802.11i, nahrazuje přechodný protokol WPA a měl by být v současnosti jediným používaným protokolem pro zabezpečení wifi sítí. Protokol WPA2 poskytuje dva módy – Enterprise a Personal (známý jako PSK; pre-shared key).

3.1.1 Personal mód (PSK)

Mód předsdíleného klíče je jednoduchý mód určený především pro zabezpečení malých domácích sítí s jedním případně několika málo přístupovými body. Sdílený klíč (nazýván též jako heslo pro přístup k síti) je uživatelem nastaven na přístupovém bodě (AP) a nastavením stejného klíče na zařízeních připojujících se k přístupovému bodu je umožněno navázání spojení s přístupovým bodem a tím přístup do sítě. Z tohoto klíče jsou následně odvozeny klíče pro danou relaci (přechodný pro unicast a skupinový klíč pro multicast a broadcast komunikaci).

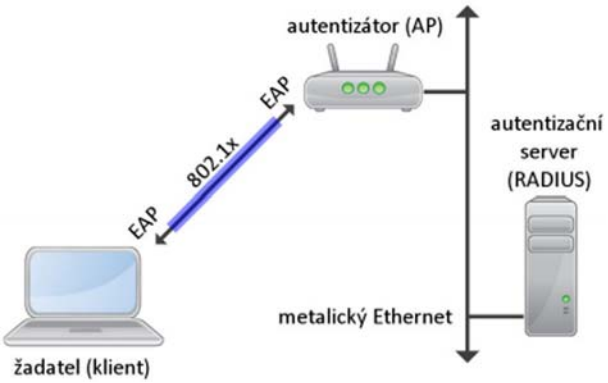
Bezpečnost v případě nasazení tohoto módu je limitována především uživatelem samotným, jelikož značně záleží na volbě předsdíleného klíče – jeho délce a náhodnosti. Uživatel může mít zpravidla tendenci zvolit zcela nevhodný klíč – ať se jedná o příliš málo znaků (většinou méně než 8 znaků) anebo slovníkovou povahu klíče (kvůli zapamatování). Výsledkem snahy o odbourání tohoto problému je eliminace možnosti volby klíče uživatelem převedená na fyzický úkon, který je potřeba provést pro vyjednání a vygenerování kvalitních klíčů mezi přístupovým bodem a uživatelským zařízením (technologie WPS).

Hlavní rysy Personal módu

- AP rozhoduje autonomně o povolení/zakázání přístupu na základě shody s nastaveným klíčem;
- není správa uživatelů, selektivní zakázání/povolení konkrétního uživatele, pro zablokování nežádoucího uživatele potřeba změnit klíč (a sdělit nový oprávněným uživatelům, kteří musí změnit své nastavení); v této souvislosti je dobré zmínit i různé nadstavby v podobě filtrování požadavků na úrovni MAC adres – nežádoucímu uživateli sice plně neznemožníme přístup do sítě (vhodnou změnou MAC adresy lze obejít), ale vyhneme se změně sdíleného klíče, která by se dotkla všech právoplatných uživatelů;
- tímto značná limitace na použití v lokální malé síti s jednotkami uživatelů, například domácí AP.

3.1.2 Enterprise mód

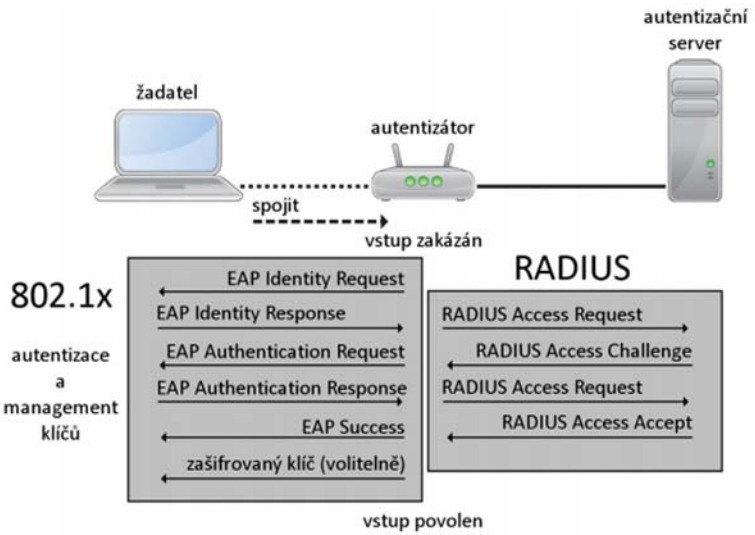
Mód Enterprise („podnikový“) je určený pro zabezpečení rozsáhlých sítí s mnoha přístupovými body (desítky, stovky), mezi kterými se předpokládá pohyb mnoha uživatelů s přenosnými zařízeními. Uživatelé mohou volně přecházet mezi přístupovými body a přístup jim je poskytnut na základě ověření pomocí tajemství, které zná uživatel a na jehož základě může být ověřen určeným způsobem (dle použité autentizační metody). Enterprise mód vyžaduje další komponenty a standardizované protokoly pro svoji funkci. K těmto patří 802.1x autentizace, RADIUS, EAP umožňující různé autentizační metody určující formu tajemství pro umožnění přístupu a vyjednání klíčů relace.



Obr. 1: Infrastruktura 802.1x. Zdroj [5]

Hlavní rysy Enterprise módu

- přítomnost autentizačního serveru a centralizovaná správa uživatelů;
- přístupový bod rozhoduje o povolení/nepovolení přístupu na základě odezvy serveru;
- uživatelé se mohou připojovat k libovolným AP v rámci infrastruktury pomocí jednoho tajemství, autentizace probíhá vůči společnému autentizačnímu serveru;
- správa a možnost kontroly přístupu pro konkrétní jednotlivé uživatele;
- závislost na dostupnosti autentizačního serveru (a páteřních sítí).



Obr. 2: Autentizace podle 802.1x. Zdroj [5]

4 Problémy s nasazením IEEE 802.11 v aplikaci FWA

Po stručném představení obou módů je patrné, že standard poskytuje nástroje pro zabezpečení wifi sítí v původně uvažovaném způsobu jejich využití – malé

domácí sítě pro propojení přenosných zařízení (Personal mód) a rozsáhlá prostředí s mnoha AP a mnoha uživateli s mnoha přenosnými zařízeními s možností pohybu po síti (Enterprise mód).

Postupem času však našly wifi sítě využití ještě v další oblasti, kde často suplují jiné technologie (především z důvodů nízké ceny v porovnání s ostatními technologiemi), a to v oblasti venkovních sítích pevného bezdrátového přístupu (FWA). Z počátku především připojováním externích antén k zařízením určeným pro klasické vnitřní nasazení, později vznikla řada výrobců specializujících se především právě na tuto oblast FWA výrobou specifických integrovaných řešení a platforem, které však mají přes své odlišnosti v softwarovém vybavení, procesoru, paměti jedno společné – a tím je rádiový čip s implementací protokolu 802.11.

Nasazení v tomto prostředí přináší určitá úskalí a omezení. První se týká protokolu samotného a způsobu detekce stavu média, kolizí apod., kde díky jiné charakteristice antén a vzdálenostech vznikají stavy, které se ve vnitřním prostředí buď neprojevují, anebo pouze omezeně. K těmto patří problém skryté a exponované stanice [4].

Tyto problémy protokolu se snaží řešit výrobci různými proprietárními úpravami protokolu a nahrazením distribuované koordinační funkce funkcí centralizovaného řízení přístupu přístupovým bodem (ovšem za cenu ztráty kompatibility mezi zařízeními různých výrobců – AirMAX, NStreme, NV2).

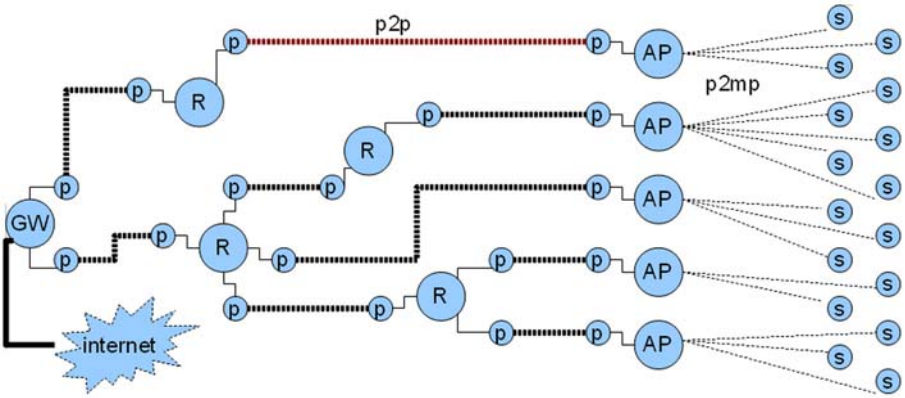
Mezi hlavní rysy wifi sítí v aplikaci FWA patří

- použití externí venkovní antény s větším ziskem a menšími vyzařovacími úhly, pevně instalované na anténním nosiči, často integrované s aktivní elektronikou v jeden celek;
- obvykle překonávané vzdálenosti od desítek metrů do kilometrů (v případě koncových stanic) a jednotek až desítek kilometrů u páteřních spojů;
- propojování vzdálených lokalit, řazení mnoha spojů za sebou – růst latencí, jitter, ztrátovost, rizika interferencí v ISM pásmech, změny parametrů přenosových tras s časem (počasí, roční období, vegetace, narušování Fresnelovy zóny, námraza, déšť aj. vlivy).

5 Zabezpečení wifi sítí v aplikaci FWA

Rozsáhlou síť tohoto typu si můžeme představit jako síť rozkládající se na geograficky značně rozlehlém území skládající se z desítek až několika set přístupových bodů umístěných obvykle na vyvýšených místech s vhodným výhledem pro pokrytí lokality signálem (vesnice, město, ...). K těmto přístupovým bodům se bezdrátově připojují jednotliví koncoví uživatelé sítě využívající služby

poskytované sítí (např. přístup k internetu). Další komponentou sítě je páteřní infrastruktura propojující přístupové body a zajišťující jejich konektivitu s dalšími sítěmi (například internet). Páteřní síť může být realizována různými způsoby – pomocí pevné sítě anebo opět bezdrátově pomocí různých pojítek obvykle typu point-to-point v licencovaných i bezlicenčních pásmech, ale velice často opět pomocí technologie založené na wifi. Postupně lze navíc očekávat migraci wifi i do dalších nestandardizovaných pásem (tj. mimo pásmo 2,4 a 5 GHz) pomocí frekvenčních konverzí (řešení již existují).



Obr. 3: Komponenty FWA sítě

Zabezpečit je tedy potřeba dvě komponenty sítě (viz obr. 3). První z nich jsou koncová AP (v obrázku označeny AP, stanice uživatele sítě jsou označeny s) a druhou je distribuční infrastruktura (v obrázku: p – konec p2p spoje, gw – hlavní brána sítě (gateway), R – retranslace). V případě AP jde o zabezpečení point-to-multipoint (p2mp) komunikace mezi AP a koncovými stanicemi uživatelů, dále je potřeba zajistit autentizaci uživatelů, žádaná je i možnost selektivně řídit přístup k síti (odpojení při nezaplacení služby, zneplatnění klíčů při odcizení zařízení apod.) U páteřní (distribuční) infrastruktury hovoříme o zabezpečení point-to-point (p2p) spojů.

5.1 Požadavky zabezpečení p2p a p2mp spojů pomocí WPA2 Personal/Enterprise

Oba již zmíněné módy protokolu WPA2 korespondují s účelem, pro který byl protokol 802.11 navržen – malé lokální sítě v rámci domácnosti pro přenosná zařízení, anebo rozsáhlejší prostředí s mnoha uživateli a přenosnými zařízeními pohybujícími se po síti s množstvím přístupových bodů připojenými do pevné

infrastruktury s možností centrální správy. Není zde mód, který by odrážel praktické nasazení sítí 802.11 v rozsáhlém prostředí složeném z bezdrátové distribuční sítě (především spojů point-to-point) a bezdrátových posledních mílí (spojů typu multipoint), navíc bez požadavku na migraci jednotlivých uživatelů mezi přístupovými body, která je zde obvykle nežádoucí.

Výchozím předpokladem v uvažované oblasti nasazení je, že by zabezpečení sítě nemělo zvyšovat riziko její nedostupnosti. Výpadek jednotlivé komponenty sítě, například páteřního spoje, by neměl způsobit rozklad zbytku sítě kvůli tomu, že není možné provést autentizaci. Tím je využití módu WPA2 Enterprise s centrálním autentizačním serverem pro celou síť se závislostí na on-line dostupnosti značně omezené. U distribučních spojů, které by měly být maximálně odolné a nezávislé, by bylo použití WPA2 Enterprise módu obtížné.

V této situaci se věnujeme využití obou módů protokolu WPA2. Je potřeba si uvědomit, jaké následky může mít nasazení konkrétní varianty na dostupnost sítě i na další bezpečnostní cíle. Z důvodů diagnostiky sítě a možnosti nalezení skutečné poruchy je potřeba zachovat co největší stabilitu. Jinými slovy zavedení zabezpečení sítě by nemělo zvýšit riziko nedostupnosti či vytvořit Single Point of Failure. Je potřeba mít na vědomí skutečnost, že síť, jejíž infrastruktura je z většiny bezdrátová a spojuje vzdálené lokality, je náchylnější na poruchy než pevná síť propojující přístupové body v jedné budově. V případě nasazení centrálního autentizačního serveru dle Enterprise modelu pak může dojít ke ztrátám v komunikaci, zvýšené latenci anebo může být komunikace přerušena úplně. Zároveň je potřeba zajistit, aby se celá síť v případě nedostupnosti autentizačního serveru postupně zcela nerozpadla kvůli nemožnosti provést autentizaci koncových stanic, případně distribučních spojů samotných. Jinak řečeno požadujeme, aby část sítě, která ztratí konektivitu, byla funkční stejně jako bez zavedení bezpečnostních mechanizmů a mohla být provedena diagnostika sítě a nalezení místa poruchy v síti.

K zabezpečení spojů p2p je možné využít mód WPA2 Personal bez dalších úprav. Spoj typu p2p je dlouhodobá záležitost, navíc zcela ve správě správce sítě. Nevyvstává zde problém potřeby zneplatňování jednotlivých stanic, potřeby obměny klíče apod. jako v prostředí p2mp se zařízeními umístěnými u koncových uživatelů. Hlavním předpokladem je vygenerování dostatečně kvalitního klíče – co se délky a náhodnosti týče a jeho jednorázové nastavení na oba konce spoje. Dále je potřeba tento klíč bezpečně uchovat pro případ výměny hardwaru jedné strany spoje (oprava poruchy) či jinou údržbu.

U p2mp spojů je situace složitější. Vedle maximální odolnosti vůči výpadkům a dalším poruchám p2p spojů zajišťujících konektivitu na p2mp AP (latence, jitter, ztrátovost, které by mohly mít vliv na úspěšnost provedení autentizace stanice) je nutné uvažovat i možnost složitějšího řízení přístupu než u p2p spojů, kde jsou pouze dvě protistrany. Jelikož p2mp připojuje k síti koncové uživatele (tj. zákazníky poskytovatele služeb), je potřeba zajistit možnost správy klíčů

a přístupu uživatelů pro jednotlivé koncové stanice. V případě nasazení Personal módu je následná obměna klíče značně komplikovaná z důvodu nutnosti změny nastavení u každé stanice. Navíc v okamžiku změny nemusí být všechny stanice dostupné (např. uživatel vypne PC vč. bezdrátové stanice), což jsou skutečnosti, které nemůže provozovatel zcela ovlivnit. Dále je potřeba uvažovat i fakt, že stanice je umístěna u samotných uživatelů, tedy potenciálně nepřátelském prostředí a bez pozornosti provozovatele mohou být na koncové zařízení prováděny útoky, včetně fyzických (rozebrání, připojení sériové konzole, nabootování z jiného média, získání obsahu konfigurace a následně klíče). Použití Personal módu tedy není řešením, často je však používán alespoň jako částečné řešení, aby byl neautorizovaný přístup k síti alespoň komplikovanější. Hlavním argumentem je ale jednoduchost (není potřeba nic krom nastavení na AP/stanicích) a splnění podmínky nezávislosti a maximální dostupnosti – pokud AP běží, stanice se může připojit bez závislosti na dalších komponentách. Posledním argumentem pro používání Personal módu je též jeho bezproblémová podpora v zařízeních všech výrobců, zatímco pro Enterprise mód jsou k dispozici různé autentizační metody a tento mód nemusí být implementován vůbec anebo mohou být implementovány metody, které se poskytovateli zrovna nehodí navíc s potřebou celé složité Enterprise architektury závislé na komunikaci s autentizačním serverem.

Z výše uvedeného vyplývá, že každý mód poskytuje jen část toho, co by bylo pro prostředí p2mp vhodné. Pro přehlednost uvádíme stručné shrnutí výhod a nevýhod jednotlivých módů.

WPA2/Personal:

+ nezávislost na dalších komponentách (tím spolehlivost a odolnost)
+ dobrá podpora ve všech zařízeních
– nemožnost selektivního řízení a zneplatňování klíčů pro jednotlivé uživatele (klíč je sdílený)
– obtížná změna sdíleného klíče (potřeba změnit u všech zařízení, některá nemusí být v okamžiku změny aktivní apod.)
– není řešena centralizovaná evidence včech klíčů použitých na síti (potřeba pro případ poruchy – výměna hw, nová konfigurace) – nutno udržovat aktuální seznamy

WPA2/Enterprise:

+ řízení přístupu na základě tajemství sdíleného s konkrétním uživatelským zařízením
+ centralizovaná správa
– závislost na on-line komunikaci s AS
– složitá infrastruktura, různé autentizační metody, potřebné nemusí být implementované

5.2 Možnosti zabezpečení p2p a p2mp spojů

Navrhujeme dva způsoby, jak se vypořádat s výše uvedenými problémy. Uvažujeme využití vlastností standardizovaných módů pro dosažení řešení. Z výše uvedeného vyplývá, že každý mód poskytuje pro uvažované nasazení dobré i špatné vlastnosti. Řešení tedy využívá dobré vlastnosti a snaží se o eliminaci špatných. Obě navrhovaná řešení vycházejí pouze z úprav softwaru přístupového bodu.

Využití Enterprise módu

Využití Enterprise módu by bylo možné spojením autentizačního serveru přímo s přístupovým bodem, tj. jednalo by se o službu běžící přímo na samotném AP. Tím by odpadla závislost na on-line dostupnosti centrálního serveru pomocí distribuční infrastruktury. Je však patrné, že tímto krokem vzniknou lokální databáze na jednotlivých AP, které je potřeba udržovat. Ostatní nevýhody týkající se podpory koncových zařízení zůstávají zachovány.

Využití Personal módu

U Personal módu je potřeba eliminovat nutnost použití sdíleného klíče. To by bylo možné takovou úpravou softwaru přístupového bodu, která by umožňovala definovat samostatný sdílený klíč pro každou jednotlivou stanici. Sdílený klíč by byl tedy sdílený pouze ve vztahu 1 : 1, tedy přístupový bod a jedna konkrétní stanice. Tím by bylo možné zneplatnit připojení konkrétní stanice např. při prozrazení klíče, či potřebě odpojit stanici (ukončený odběr služby apod.). Ostatní vlastnosti tohoto módu by zůstaly zachovány.

Enterprise + Personal

Nastíněné řešení využití obou módů v uvažovaném prostředí řeší vytyčené cíle pro:

1. zajištění maximální robustnosti a nezávislosti na on-line komunikaci s dalšími komponentami při nasazení zabezpečení
2. využití dostupných módů s minimem zásahů.

Otázkou zůstává centralizovaná správa klíčů rozmístěných v lokálních databázích. Za tímto účelem je potřeba navrhnout infrastrukturu pro centralizovanou správu a distribuci klíčů (aktualizace místních databází) na přístupových bodech.

V této souvislosti je potřeba zajistit:

- vygenerování vhodných klíčů (délka, náhodnost)
- zajistit jejich bezpečnou distribuci na koncový AP (zabezpečené spojení, autentizace) – aktualizace místní databáze (přidání/zneplatnění) – provádí se pouze v případě změny

- vytvořit centralizovanou zabezpečenou databázi s vhodným rozhraním (web) umožňující správcům sítě vést na jednom místě evidenci všech uživatelů sítě a AP a v případě úprav (přidání, odebrání, ...) provést automaticky aktualizaci místních databází.

6 Shrnutí a závěr

V naší práci jsme se pokusili nastínit problémy se zabezpečením bezdrátových sítí v aplikaci FWA a možnosti vedoucí k řešení současných problémů. Popsali jsme několik kroků potřebných ke vzniku systému vhodného pro správu FWA sítí. Zabezpečení těchto sítí je dnes často řešeno nedostatečně, protože poskytovatel se musí vypořádat s mnoha dalšími aspekty provozu sítě (např. řízení šířky pásma, výběr a rozmístění antén, eliminace zarušení, navyšování kapacity), takže nasazení čehokoliv, co by dále zkomplikovalo správu sítě anebo zvýšilo riziko provozních problémů (nedostupnost), je přijímáno s obavami, příp. raději nenasazeno i za cenu připuštění možnosti neoprávněného využití služby. Vyvstává též otázka poměru mezi vynaloženým úsilím do kvalitního zabezpečení sítě a přínosem (ztráty způsobené neoprávněným využitím/zneužitím).

Poděkování

Tato práce byla podporována FIT VUT v rámci grantu Pokročilé bezpečné, spolehlivé a adaptivní IT (FIT-S-11-1) a výzkumným záměrem Výzkum informačních technologií z hlediska bezpečnosti, MSM0021630528.

Literatura

[1] Singh, T., Verma, A. K.: Security Aspects of IEEE 802.11a and HiperLAN/2. In *Proceedings of National Conference on Challenges & Opportunities in Information Technology*, 2007 (COIT-2007).

[2] *Security and the Motorola Canopy Wireless Broadband Platform*, Motorola Inc., 2003.

[3] Trimintzios, P., Georgiou, G.: WiFi and Wimax Secure Deployments. In *Journal of Computer Systems, Networks and Communications*, Vol. 2010.

[4] Liu, Ch.: *A Survey of the Hidden Station Problem in Wireless Networks*, Project of CPRE543, dostupné online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.4398&rep=rep1&type=pdf

[5] Pech, M.: *Problematika bezdrátových sítí*, bakalářská práce, FP VUT, 2011, dostupné online: http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=37419

IPsec v otevřených systémech

Pavel Šimerda

E-mail: pavlix@pavlix.net

1 IPsec na Linuxu

S protokolem IPv6 přišel do světa TCP/IP standard pro zabezpečení komunikace, IP security, kterému nakonec zůstalo zkrácené jméno IPsec.

IPsec lze použít k ochraně komunikace na síťové vrstvě, tedy zcela nezávisle na transportních a aplikačních protokolech a stejně nezávisle na linkové technologii. Liší se tak od mnoha stávajících řešení pro zabezpečenou komunikaci.

Je primárně navržený pro globální IPv6 síť bez IP maškarády, ale byl postupně doplněn o možnost provozu ve starších sítích. Ať už se použije kdekoli, jeho cílem je rozšířit protokol IP o autentizaci, integritu a utajení.

1.1 IPsec aneb zabezpečení na síťové vrstvě

Jednou z nejvýznamnějších vlastností IPsecu je jeho flexibilita. Lze použít v mnoha běžných scénářích zabezpečené komunikace. Mezi nejčastější použití patří propojovací VPN mezi sítěmi a zabezpečené připojení notebooků do firemní sítě. Ona samotná zkratka VPN naznačuje, že vytvoříme zabezpečenou virtuální síť, která bude fungovat napříč Internetem.

S globálními IPv6 adresami není třeba ani sahat po virtuálních sítích s vlastní adresací. Stačí skombinovat globální adresaci IPv6 Internetu se zabezpečením, které nabízí IPsec. Zabezpečenou komunikaci tak lze provozovat úplně bez zásahu do směrování, což vede na oddělení těchto dvou oblastí.

Kouzelné slovo VPN ale není jediným případem užití. IPsec může v mnoha případech nahradit zabezpečení transportní vrstvy jako je TLS/SSL nebo zabezpečení linkové vrstvy jako je WPA2. Na použití IPsec místo TLS nejsou ale připravené aplikace. Co se z toho všeho vyvine, ukáže až čas (a může to být čas dost dlouhý).

1.1.1 Protokol ESP

IPsec nabízí na ochranu komunikace protokoly ESP a AH. Zatímco ESP nabízí autentizaci, integritu a utajení, AH vypadá jako takový chudší bratranec, který zajišťuje pouze autentizaci a integritu.

Autentizací se v tomto případě myslí ověření odesilatele paketu na základě dohodnuté bezpečnostní asociace. Význam integrity a utajení je v tomto případě zřejmý, tyto dva pojmy dohromady znamenají, že pakety přijdou od odesilatele v nezměněné podobě a že si obsah komunikace nepřečte někdo nepovolaný.

Častěji tak narazíte na zajímavější ESP, tedy Encapsulated Security Payload, což neznamená nic jiného než zašifrovaná data zabalená do IPv4 nebo IPv6 paketu. Číslo protokolu ESP je 50 a v klasickém IPsecu hlavička ESP následuje IP hlavičku.

Pro názornou představu doporučuju představovat si hlavičky jednu podruhé, následované aplikačními daty, i když je ve skutečnosti obsah ESP paketu zašifrovaný, takže další hlavičky nejsou vidět.

1.1.2 Tunel nebo transport

IPsec ESP používá dva naprosto odlišné přístupy, které se dělí podle toho, jestli se za některým z koncových bodů bezpečnostní asociace nachází další počítačová síť, kterou dotyčný bod zpřístupňuje.

Jednoduššímu případu, kdy koncové body bezpečnostní asociace jsou zároveň koncovými body komunikace, se říká transportní mód. ESP paket v transportním módu obsahuje pouze jednu IP hlavičku, a to tu, která předchází ESP, tedy je nešifrovaná. Znamená to, že se stejné IP adresy používají pro bezpečnostní asociaci jako pro komunikaci samotnou. Za hlavičkou ESP, v šifrované části, pak obvykle následuje hlavička transportního protokolu jako je UDP nebo TCP.

Složitější tunelový mód se používá na pro případ, kdy nestačí jedna zdrojová a jedna cílová adresa, ale je potřeba rozlišit adresy koncových bodů komunikace od adres koncových bodů bezpečnostních asociací. To se u IPsecu děje ve dvou případech. Jedním je koncová VPN brána, který prostřednictvím tunelu zpřístupňuje celou nějakou síť, druhým je cestovní VPN klient na IPv4 síti za NATem, který pro správnou funkci dostane od VPN brány přidělenou adresu pro virtuální rozhraní. Je tedy pokrytý jak případ propojení celých sítí, tak případ připojení jednoho počítače do sítě zvenčí.

Technicky je tunelový mód řešen velmi jednoduše. V šifrované části ESP paketu se nachází druhá IP hlavička, ve které je místo pro další zdrojovou a cílovou IP adresu. Možná by se to dalo přirovnat ke čtyřadresnému formátu rámců bezdrátového Ethernetu při použití WDS. Vnější, nešifrovaná IP adresa je určená ke směrování mezi koncovými body takto vytvořeného tunelu (proto tunelový mód), vnitřní, skrytá, IP adresa slouží ke směrování před tunelem a za tunelem.

Tím pádem tunelový mód před světem skrývá dokonce i identitu konkrétních komunikujících počítačů, jediné, co je vidět, je že probíhá komunikace mezi sítěmi.

Nikde není psáno, že se dvojice zdrojových adres nebo dvojice cílových adres nemůže při komunikaci v tunelovém módu shodovat. Nicméně, pokud by se měly shodovat ve veškeré komunikace proudící v rámci této bezpečností asociace, nabízí se možnost nahradit tunelový mód efektivnějším a jednodušším transportním módem. Zatím se obecně držím pravidla, že transportní mód používám pro bezpečnostní asociace mezi koncovými body komunikace a tunelový mód používám pro všechny ostatní případy.

1.1.3 IPv6 versus IPv4 maškaráda

Ač se ve většině případů odlišnosti IPv6 a IPv4 tolik neprojevují, v případě IPsecu hraje roli přítomnost či absence IP maškarády. Zatímco na IPv6 se zabezpečení (IPsec i firewall) dá řešit zcela nezávisle od směrování a propojování sítí, IPv4 si vynucuje trochu větší péči.

Na IPv6 se člověk může vždy rozhodnout, jestli použije IPsec v transportním módu mezi koncovými stroji nebo v tunelovém módu mezi bránami sítí, u IPv4 bývá potřeba zajistit unikátnost síťových rozsahů a jejich propojení. Propojení lze zajistit jednoduchým nešifrovaným tunelem anebo rovnou použít IPsec v tunelovém módu.

Tunely mezi IPv4 sítěmi se navíc navazují pomocí veřejných IPv4 adres, mezi kterými je možné oboustranně navazovat spojení. V tunelovém módu se takové propojení chová stejně jako v IPv6.

1.1.4 IKE

Autentizace, šifry, klíče a další parametry propojení se domlouvají pomocí protokolů IKEv1, IKEv2 nebo alternativ. Obě verze protokolu IKE se přenášejí transportním protokolem UDP a standardně sídlí na portu 500.

1.1.5 IPv4 NAT-Traversal

IPsec se na IPv4 často používá k připojování cestovních klientů za IP maškarádou k VPN bráně. Tím, že se ESP vkládá přímo do IP paketů, tak není zdaleka samozřejmostí, že ESP pakety budou procházet NATem. Navíc vznikají problémy i s UDP komunikací na portu 500, a to kvůli snahám některých vylepšených routerů speciálně naložit s pakety protokolu IKE.

Řešení jménem NAT-T spočívá v tom, že se IKE při detekci IP maškarády (nebo pevnou konfigurací) přesouvá z problémového portu 500 na k tomu určený port 4500. A následně se do UDP paketů na portu 4500 schová i celý ESP provoz. Navenek tedy IPsec vypadá jako úplně obyčejná UDP VPN na portu 4500.

Alespoň takto to popisují příslušná RFC. Nikde se nepíše, že nelze celý systém přizpůsobit a například v kombinaci s ICE (obecný standard pro procházení UDP provozu NATem) navazovat přímé šifrované tunely i mezi stroji za IPv4 maškarádou. Stejně tak nikdo neříká, že nelze NAT-T použít pro kteroukoli jinou situaci včetně překonávání stavových firewallů nebo že se musí dodržet známá čísla portů.

1.2 IPsec na Linuxu

A teď přejdeme k té praktičtější, zajímavější, ale také pesimističtější části. Protokol IPsec je modulární, spousta jeho vlastností je nezávislá na těch ostatních, ale problém je, jak si s tím vším poradí implementace.

Implementace IPsecu na Linuxu, teď mám na mysli jako celek, má k dokonalosti daleko, ale to stejné se týká i ostatních platforem. Ostatně poznáte dále. Jaderná část IPsecu na mě působí velmi dobrým dojmem, ale userspace nástroje jsou spíše neohrabané. Různým implementacím chybí různé důležité vlastnosti. Chybí jednoduché ovládání pro běžné uživatele IPsec VPN. Celý ekosystém působí spíše zmateným dojmem.

Linuxová implementace, jak je dobrým zvykem, funguje jako skládačka. Z malých stavebních prvků můžete postavit téměř libovolné řešení. Při troše šikovnosti se můžete bezpečně připojovat k desítkám služeb i VPN, aniž by se tyto ovlivňovaly mezi sebou. Budoucnost zabezpečení sítí pravděpodobně patří IPsecu a tak se bude ledacos měnit i v linuxových distribucích, snad k lepšímu. Dál se podíváme, co taková skládačka nabízí.

1.2.1 IPsec v jádře

Implementace IPsecu v linuxovém jádře zahrnuje vedle samotné kryptografie a zpracovávání paketů i správu bezpečnostní asociací a bezpečnostních politik. To mimojiné znamená, že lze IPsec zkoušet bez jakýchkoli běžících démonů v uživatelském prostoru.

Jediné, co potřebujete k úspěšnému otestování IPsecu, je vytvořit odpovídající záznamy v databázi bezpečnostích asociací a databázi bezpečnostních politik. Podle bezpečnostní politiky jádro rozhoduje, kterých paketů se týká zabezpečení a jakým způsobem se má provádět. Bezpečnostní asociace poskytuje konkrétní šifry a klíče.

1.3 Příkaz setkey

K vyzkoušení vystačíte s příkazem setkey, který bývá v balíku `ipsec-tools`. Spustit pomocí setkey šifrovaný kanál je velice jednoduché, stačí použít direktivy `add` a `spdadd`.

Následující direktivy například spouštějí jednosměrný šifrovaný kanál z IPv4 adresy 192.168.1.2 na IPv4 adresu 192.168.3.4:

```
add 192.168.1.2 192.168.3.4 esp 0x1 -E rijndael-cbc
        0x3ed0af408cf5dcbf5d5d9a5fa806b224;
spdadd 192.168.1.2 192.168.3.4 any -P out
        ipsec esp/transport//require;
```

Konfigurace je na obou koncích stejná, jen u direktivy spdadd směr `out` (ven) nahradíte na druhé straně za směr `in` (dovnitř). Pomocí pingu a tcpdumpu můžete ověřit i to, že komunikace bude směrem od `192.168.1.2` k `192.168.3.4` probíhat šifrovaně. V praxi byste pak zprovoznili i opačný směr. Detailní popis syntaxe najdete v manuálové stránkce k příkazu `setkey`. Snad jen, `0x1` je SPI, číslo které má rozlišit různé klíče pro stejný cíl, `rijndael-cbc` jsem použil kvůli tomu, že setkey odmítal rozumět `aes-cbc`, ale stejně dobře můžete používat i `3des-cbc`.

Linuxové distribuce obvykle obsahují pro setkey initskripty, které zajišťují načtení konfigurace IPsecu při bootu a při případném dalším spuštění initskriptu. Snad poslední poznámka, abyste se nespálili. Linuxový IPsec se chová mírně odlišně od toho z BSD a setkey naopak emuluje chování z BSD tím, že přidává do tabulky bezpečnostních politik některé záznamy navíc. Dá se to obejít přepnutím setkey do jaderného módu volbou -k, což způsobí, že setkey nedělá nic navíc.

Dokonce se `setkey` můžete vyhnout úplně, využít standardního příkazu `ip xfrm` a initskripty si zajistit sami:

```
# ip xfrm state add src 192.168.1.2 dst 192.168.3.4 \
proto esp spi 1 enc aes 0x3ed0af408cf5dcbf5d5d9a5fa806b224
# ip xfrm policy add dir out src 192.168.1.2 dst 192.168.3.4 \
tmpl proto esp mode transport
```

1.4 Démon jménem Racoon

Racoon, česky ämývalô, se stará o domluvu klíčů pomocí protokolu IKEv1. Pochází z projektu KAME, stejně jako ostatní součásti `ipsec-tools`. Jeho hlavní nevýhodou je, že podporuje pouze IKEv1, ale pro běžnou práci se sítí to stačí.

Konfigurace IKE démona spočívá v nastavení, jak si má s ostatními domlouvat bezpečnostní asociace. Pro každý stroj, se kterým má Racoon navazovat kontakt, se nastavuje způsob autentizace a šifrování protokolu IKE. Další konfigurace obsahuje informace o domlouvaných bezpečnostních asociacích, tentokrát pochopitelně bez klíčů.

Zajímavý je způsob, jakým se Racoon zapojuje do práce jádra. IPsec se dá používat zcela automaticky. Stačí mít správně definované bezpečností politiky

(pomocí setkey) a jádro samo kontaktuje běžící Racoon, aby domluvil bezpečnostní asociace pro právě zahajovanou komunikaci. Snad jediná vada na kráse je, že se mi nepodařilo zbavit výpadku několika paketů během zahajování komunikace.

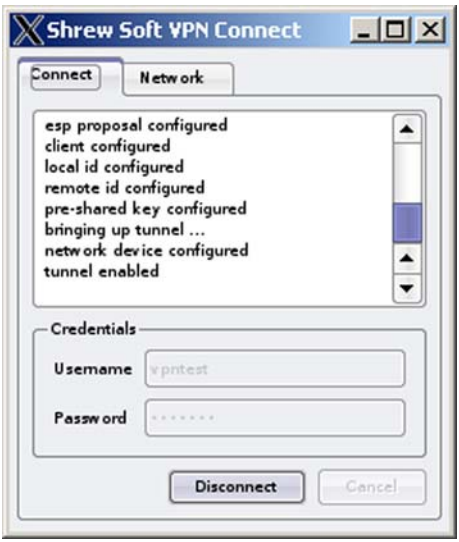
Opačným přístupem je použít `racoonctl` (nebo na Debianu bohatší alternativu `racoon-tool` s vlastní konfigurací), kdy bezpečnostní politiky nenastavíte a VPN aktivně spustíte, přičemž Racoon sám politiky připraví, případně pustí vámi nakonfigurovaný skript, který může na úspěšnou domluvu bezpečnostích asociací reagovat.

Pro konkrétní příklady konfigurace si vás dovolím odkázat na můj oblíbený zdroj praktických ukázek na http://www.ipsec-howto.org/. Není kompletní, ale to nejdůležitější tam najdete.

1.4.1 Alternativy

Vedle Racoonu najdete další, obvykle méně zdokumentované a méně zavedené alternativy. Asi nejznámější je Openswan, který má poněkud chudou dokumentaci, takže se těžko ověřuje, co vlastně umí. O poznání zajímavěji vypadá strongSwan, který v implementovaných vlastnostech zřejmě dalece předčí jak Racoon, tak i Openswan, příkladem budiž protokol IKEv2. Další plnohodnotnou implementací by mohl být Racoon2, ale projekt se zdá být relativně tichý a v distribucích nejsou balíčky, takže těžko říct.

Z GUI programů stojí za zmínku Shrew Soft VPN Client pro Linux a BSD, který vypadá použitelně:



A samozřejmě se nesmí zapomenout na strongSwan plugin do všudypřítomného NetworkManageru:

Celkově z toho mám dojem, že očekávané zvýšení zájmu o IPsec přinese určité vykrystalizování implementací IKE a dalších nástrojů.

1.4.2 Firewall

Možnost kombinovat IPsec a firewall je naprosto klíčová. Zde stojí za zmínku fakt, že IPsec na Linuxu nevytváří virtuální rozhraní, na které by se dal vztáhnout Firewall. Místo toho pakety prochází firewallem jak v šifrované, tak i v čitelné podobě a iptables obsahují modul `policy`, pomocí kterého můžete u paketů v čitelné podobě kontrolovat jejich stav z hlediska zabezpečení při průchodu sítí.

Následující jednoduchá pravidla například povolí protokol ESP a všechen provoz, který přichází šifrovaně přes IPsec.

```
iptables -A INPUT -p esp -j ACCEPT
iptables -A INPUT -m policy --pol ipsec --dir in --proto esp
         -j ACCEPT
```

1.5 Případ užití: SSH přístup

Jako bonus bych vám rád nabídl nápad, na který mě přivedla správa cizích serverů po SSH. Jednou z nejčastějších technik äzabezpečeníô (já tomu říkám spíše udržování přehledných logů) je přesunutí SSH na nějaký alternativní port, většinou dávno profláknutý. Dalším krokem bývá omezení zdrojových adres pro připojení k SSH. Obě tyto metody jsou svým způsobem otravné a obvykle je obcházím vyladěním souboru `.ssh/config` a připojováním přes jiný SSH server.

Hraní s IPsecem mě přivedlo na nápad, že bych se mohl vhodnou kombinací firewallu a IPsecu tomuto všemu vyhnout tím, že nastavím bezpečností politiky na klientovi, aby se na server vždy připojoval přes IPsec (ať už přímo, nebo třeba přes jeden stroj, který pak slouží de facto jako VPN gateway). Rozdíl od klasických VPN je ten, že tuto politiku můžu mít nastavenou trvale, aniž by zasahovala do ostatní komunikace. Při pokusu o připojení má IKE démon na serveru možnost reagovat na změnu mojí IP adresy a vygenerovat odpovídající bezpečnostní politiku. Pak už zbývá jen zajistit, aby server doručil pakety na cílový stroj (pokud tím cílovým strojem není sám).

Toto řešení s sebou nese mnohé výhody oproti dohadování se, jestli se na SSH zakáže přihlašování heslem a dalším problémům spojeným s nefiltrovaným provozem SSH, a dobře se kombinuje se stávajícím řešením typu povolení jen z některých IP adres. Podobných případů užití jistě najdete spoustu třeba i u různých webových administrací a dalších služeb, které z praktických důvodů nechcete nebo nemůžete zabezpečit na aplikační úrovni.

Šifrování disků (nejen) v Linuxu

Milan Brož

E-mail: mbroz@redhat.com

Abstrakt

Příspěvek popisuje systémy používané pro softwarové šifrování disků v Linuxu včetně jejich speciálních vlastností a rozšíření.

1 Úvod

Většina pojednání o šifrování začíná úvodem, v němž se lze dočíst, jak se šifrování stalo důležitou součástí IT počínaje vládními institucemi, velkými korporacemi a konče aktivisty a jednotlivci.

Ve skutečnosti se možnosti nasazení (zejména dostupnost speciálního hardware) u těchto skupin natolik liší, že je lze jen těžko srovnávat.

Přesto je možné s open-source nástroji, minimálními náklady a důsledným dodržováním jednoduchých zásad dosáhnout velmi rozumného kompromisu.

Ale také existuje mnoho systémů, které se spíše zaměřují na divadlo, které vyvolá dojem falešného bezpečí, než na řešení samotné.

Následující text je pokusem o pragmatický pohled na konkrétní problematiku šifrování disků, nikoliv přesné teoretické pojednání.

Nejsou zde popsány další nutná opatření, počínaje řešením incidentů a důsledným dodržováním životního cyklu hardware (ale i dat). A navíc, nejslabším článkem zůstává většinou člověk.

2 Šifrování disku

Šifrování dat lze provádět na libovolné vrstvě, počínaje aplikací, souborovým systémem, diskem nebo také může být šifrován komunikační kanál k datovému úložišti.

Posouzení, kterou vrstvu použít, je nad rámec toho pojednání.

Některé výhody a nevýhody softwarového šifrování disku na úrovni sektorů:

+ transparentní (vyšší vrstva nemusí o šifrování vědět)
+ lze použít souborový systém dle vlastního výběru
+ rozhodnutí, která data šifrovat není na uživateli
+ lze použít pro swap a hibernaci
+ odstranění klíče stačí k destrukci dat (secure disposal)
+ vhodné pro zabezpečení notebooků či přenosných datových uložišť
− problémy s víceuživatelským přístupem (jen na úrovni více hesel)
− únik klíče znamená únik všech dat
− není odolné proti útokům na standardní hw (lze je jen omezit)
− v určitých případech má znatelný vliv na výkonnost systému

Nebudeme se zde zabývat šifrováním disku na úrovni hardware, ale většina problémů zde platí taktéž.

2.1 Disk a sektor

V unixové terminologii je *blokové zařízení* definováno jako typ zařízení, kde atomickou jednotkou pro přístup k datům je blok.

Z pohledu hardware je blok nazýván *sektor*. Protože blokové šifrovací algoritmy používají blok v jiném významu, dále se v textu setkáte v tomto významu jen s pojmem sektor.

Sektor je historicky 512 bytů, u diskových polí a novějších disků je použitý sektor 4 096 bytů. Většina šifrovacích nástrojů však interně používá sektor velikosti 512 bytů.

Šifrování na úrovni sektorů znamená, že *každý sektor disku je šifrován nezávisle*. Výsledkem je tedy zase disk o stejném počtu sektorů a stejné velikosti sektoru.

Tato logika umožňuje vložit šifrování mezi souborový systém a vlastní disk, bez nutnosti dalších změn, vrstva je tedy transparentní pro uživatele nad ní.

Disk (původní) obsahující šifrované sektory budeme nazývat *ciphertext zařízení*, virtuální disk nad ním obsahující sektory v otevřené podobě *plaintext zařízení*.

Zápisy a čtení plaintext disku se transparentně šifrují/dešifrují na ciphertext zařízení.

3 Algoritmy

Všechny ilustrační příklady jsou založeny na reálných aplikacích

- **dm-crypt/LUKS** – nativní Linuxové řešení [1]
- **loop-AES** – alternativní implementace [2]
- **Truecrypt** – multiplatformní řešení [3]
- **BitLocker** – komerční řešení ve Windows Ultimate [4]

Existuje nepřeberné množství dalších implementací [5].

3.1 Blokové šifry

Algoritmy používané pro šifrování disku musí být dostatečně rychlé, aby zvládaly stejnou propustnost jako nešifrované zařízení a zároveň musí umožnit nezávislé šifrování sektorů.

Reálně tyto požadavky splňuji blokové šifrovací algoritmy (k dešifrování je třeba tajného klíče). Velikost klíče bývá obvykle 128 nebo 256 bitů. Některé systémy používají více klíčů (loop-AES používá 64 klíčů, které podle čísla sektoru rotuje).

Příkladem používaných blokových šifer pro šifrování disků jsou **AES** (Rijndael), **Twofish** a **Serpent** (a mnoho jiných).

3.2 Inicializační vektor

Pro dodržení podmínky, že stejná data v různých sektorech musí mít rozdílný ciphertext, je nutné, aby použitý šifrovací mód šel inicializovat pro každý sektor zvlášť (tweakable mód). Tato inicializační (počáteční) hodnota se obvykle nazývá *inicializační vektor (IV)*.

Do obsahu sektorů není možné uložit žádnou další informaci (jednoduše na ni není místo, neboť ciphertext a plaintext má stejnu velikost) IV je tedy obvykle odvozen od čísla sektoru na disku (offset) a případně i tajného klíče.

Inicializační vektor může být *offset* (číslo sektoru) nebo *funkce*, která zpracovává číslo sektoru a klíč (např. *ESSIV* – Encrypted Salt-sector IV [7]).

3.3 Šifrovací mód

Blokové šifry zpracovávají vždy jeden celý blok dat, typicky je blok 16 bytů (128 bitů). Diskový sektor je však větší – 512 bytů. Jeho zpracování je tedy potřeba rozdělit na jednotlivé bloky, šifra pak pracuje v *módu*, který tyto bloky zpracovává postupně.

Příkladem blokových módů je

- **CBC** (Cipher Block Chaining). CBC je klasickým módem. Je třeba podotknout, že je nutné použít inicializační vektor, který není predikovatelný.

- **XTS** (založený na módu XEX – XOR Encrypt XOR). XTS je relativně novým módem, mimo jiné řeší problém predikovatelného IV, takže pro IV lze použít přímo číslo sektoru. XTS interně používá dva klíče (512 bit klíč tedy znamená 2×256 bit klíč.)

Někdy je před vlastním šifrováním nad daty provedena oprace, která má odstranit známé nedostatky šifrovacího módu (zejména CBC), příkladem je difúzní algoritmus nazvaný *Elephant* používaný pro BitLocker.

3.4 Příklad – dm-crypt

Se znalostmi uvedenými výše již lze snadno pochopit, co znamenají následující specifikace používané pro dm-crypt [7].

- **aes-xts-plain64** – AES v XTS módu, IV je číslo sektoru

- **aes-cbc-essiv:sha256** – AES v módu CBC, IV je ESSIV/sha256

4 Správa klíčů

Bezpečnost celého šifrování závisí na kvalitě klíče a jeho bezpečném uložení.

4.1 Generování klíče

Při inicializaci je proto kritické, jakým způsobem se klíč generuje.

Uživatel si nepamatuje vlastní klíč, ale heslo (passphrase) nebo PIN, pomocí kterého se ke klíči dostane.

Vlastní klíč se při inicializaci generuje pomocí **generátoru náhodných čísel** (RNG – Random Number Generator). Často je vyžadována nějaká náhodná činnost uživatele (vstup z klávesnice, pohyb myši) která zaručí čerstvý přísun náhodných dat pro inicializaci RNG.

Klíč lze také přímo odvodit od hesla (například pomocí algoritmů **PBKDF2** (Password Based Key Derivation Function).

Základní princip většiny systémů se však drží pravidla, že *klíč na hesle přímo nezávisí a je náhodný.*

4.2 Uložení klíče

Klíč je pak možné uložit na externím tokenu, specializovaném úložišti, na dedikované šifrované části disku (někdy odděleně od vlastních dat) či v souboru chráněném heslem.

Příklady (zjednodušeně):

- Truecrypt – klíč je uložen na stejném disku jako data, šifrován heslem s přidanou solí (sůl je jediná viditelná část na ciphertext zařízení, vlastní Truecrypt hlavička je šifrována).
- loop-AES – klíče jsou uloženy v souboru, jenž je šifrován pomocí GPG.
- LUKS/dmcrypt – parametry šifrování (algoritmus, délka klíče, sůl) jsou uloženy ve viditelné hlavičce, vlastní klíč je uložen šifrovaný v dedikované oblasti (keyslot).
- BitLocker – kombinace, klíč může být uložen v TPM (viz dále), ale i šifrovaně na disku.

4.3 Mazání klíče

Bezpečnost závisí na tajném klíči, odstranění klíče znamená v principu bezpečné zničení dat.

Některé systémy se při mazání klíče snaží počítat s vlastnostmi disků (relokace sektorů v rámci disku, wear leveling).

Příkladem je anti-forensic filter pro LUKS, který alokuje kesylot a ukládá jeho obsah takovým způsobem, že i při aktivní realokaci části keyslotu v rámci firmware disku se z kopie sektorů nedá klíč zrekonstruovat.

4.4 Obnova klíče

Porucha disku (v oblasti s uloženým klíčem) či tokenu nebo TPM znamená totální ztrátu dat.

Proto se některé systémy pokouší tuto situaci dopředu řešit.

- Truecrypt – zálohou hlavičky (na disku je druhá kopie), recovery CD
- BitLocker – mnoho mechanismů, některé nejsou přesně zdokumentované
- LUKS – záloha hlavičky do souboru

Špatně navržené mechanismy obnovy mohou naprosto degradovat bezpečnost celého řešení.

5 Výkonnost

Bezpečnost nikdy není zadarmo. V případě šifrování disku nás stojí minimálně procesorový čas. Vzhledem k rychlosti moderních procesorů a disků jde o stálý zápas, která část se stává úzkým hrdlem. V minulosti to byl často disk (propustnost disků v noteboocích nebyla závratná), s dnešními SSD disky se problém spíše přesouvá na stranu procesoru.

Od prvopočátku se objevují pokusy o specializovanou akceleraci (pominemeli speciální oddělené kryptoprocesory a akcelerační karty), objevují se akcelerátory šifrování na běžných procesorech (AMD Geode, VIA Padlock, sada instrukcí AES-NI v procesorech Intel, totéž v nových procesorech AMD).

Další možností urychlení je paralelní výpočet. Například XTS mód je navržen tak, že umožňuje paralelní zpracování bloků.

Další urychlení je možné pomocí paralelního zpracování sektorů či celých IO sekvencí na více-jádrových procesorech.

6 LUKS a dm-crypt

6.1 cryptsetup

Nativní Linuxové řešení pro šifrování disku se skládá ze dvou částí. Jaderná část je **dm-crypt** a zajišťuje vlastní šifrování disku. Správa a bepečné uložení klíče je pak zajištěno pomocí aplikace **cryptsetup** v uživatelském prostoru. Cryptsetup používá formát **LUKS** pro uložení klíčů a pro jednoduchou přenositelnost šifrovaného disku mezi systémy.

Dm-crypt umožňuje využívat všechny šifrovací algoritmy a podporované hardwarové systémy v rámci jaderného cryptoAPI [7].

6.2 LUKS (Linux Unified Key Setup)

LUKS [1] zajišťuje standardní formát pro šifrovaný disk a bezpečné uložení klíčů na disku.

První vlastností je *podpora více klíčů* a jejich správu (je tedy možné přidělit více rozdílných hesel, například různým uživatelům, případně je odebrat bez nutnosti přešifrování celého disku nebo vyzrazení jiného klíče).

Klíč je uložen v **keyslotech**, které jsou šifrovány stejným algoritmem jako data, jen se klíč generuje pomocí PBKDF2 s velkým počtem iterací (časově náročný výpočet iterací je ochrana proti slovníkovému útoku). Standardní počet iterací se určuje na systému, kde je disk formátován, tak, aby trval cca 1 sekundu. Hlavička obsahuje kontrolní součet (který používá v prncipu stejný algoritmus jako keyslot), lze tedy po provedení iterace určit, zda dešifrovaný klíč je v pořádku (před jeho použitím pro reálný disk).

Další vlastnost je, že LUKS *poskytuje ochranu vůči obnovení již jednou smazaného klíče* forenzní analýzou disku, například v případě automatické realokace vadného sektoru samotným diskem. Takovýto sektor již není možné standardními metodami přepsat, a pokud by obsahoval například klíč, mohlo by dojít k obnovení informace, která měla být již vymazána. LUKS zajišťuje, že citlivé informace jsou rozloženy na disku tak, že pravděpodobnost takovéhoto útoku je minimalizována, resp. že spolehlivé odstranění byť jen části uchovaného klíče znemožní jeho rekonstrukci.

LUKS hlavička, obsahující metadata pro správu klíčů a parametry použité pro šifrované zařízení je uložena na počátku disku.

6.3 Práce s LUKS diskem

Většina distribucí již podporuje nastavení šifrování přímo při instalaci systému.

- **cryptsetup luksFormat DEV** – vytvoření LUKS a klíče (s defaultními parametry)
- **cryptsetup luksOpen DEV DEV_crypt** – zpřístupnění šifrovaného zařízení
- **cryptsetup luksClose DEV_crypt** – zrušení mapování

Příklad nakonfigurovaného disku může pak vypadat například takto:

```
# cryptsetup status sda2_crypt
/dev/mapper/sda2_crypt is active and is in use.
  type:    LUKS1
  cipher:  aes-xts-plain64
  keysize: 256 bits
  device:  /dev/sda2
  offset:  8192 sectors
  size:    184147968 sectors
  mode:    read/write
```

6.4 LUKS + LVM

LUKS/dm-crypt lze kombinovat s LVM (Logical Volume Management). Není třeba se nechat odradit komplexností LVM, v tomto případě se používají jen základní vlastnosti a celá konfigurace je triviální.

Pro nasazení na notebooku je zřejmě ideální šifrování celou diskovou oblast a nad tímto diskem pak vytvořit pomocí LVM jednotlivé diskové svazky.

```
NAME                        FSTYPE      MOUNTPOINT
sda
|-sda1                      ext3        /boot
|-sda2                      crypto_LUKS
| `-sda2_crypt (dm-0)       LVM2_member
|   |-notebook-data (dm-1)  ext4        /home
|   |-notebook-swap (dm-2)  swap        [SWAP]
|   `-notebook-root (dm-3)  ext4        /
`-sda3                      ext4        /test
```

Pro uživatele *btrfs* je situace ekvivalentní (místo LVM a jiného souborového systému je nad dm-crypt zařízením btrfs a v něm svazky, snímky apod.)

7 Zajímavé problémy

7.1 Skrytý disk

Vzhledem k vlastnostem šifrování, ciphertext (bez znalosti klíče) by neměl být rozpoznatelný od náhodných dat. Náhodná data na disku vzbudí důvodné podezření, že disk obsahuje šifrovanou část.

Na této vlastnosti je postavena implementace **skrytého disku** (hidden disk). Existují dvě mapování – jedno falešné, které po zadání hesla odkryje neškodná data a další, které v oblasti nepoužívané falešným mapováním ukládá skutečná data, která mají zůstat utajena. Do skryté části lze ukrýt i celý operační systém.

Příkladem implementace je skrytý disk v Truecryptu. V Linuxu lze podobného efektu dosáhnout mapováním dmcrypt zařízení.

Technická realizace je jedna věc, praktické důsledky (zejména v zemích, kde je odepření vydání hesla trestným činem) je věc druhá. Zda-li takové řešení uspěje při cílené prohlídce, kde do hry vstupují psychologické aspekty výslechu se většinou již nerozebírá.

7.2 SSD a TRIM

SSD (Solid State Drive) velmi často používají **TRIM** příkaz, kterým vyšší vrstva (např. souborový systém) informuje disk, že dané sektory nejsou používány (a zařízení nemusí nadále udržovat datový obsah těchto sektorů). Sektory uvolněné pomocí TRIM jsou snadno detekovatelné (ve většině zařízení obsahují samé nuly).

U šifrovaného disku tato informace může otevřít vedlejší kanál, kterým mohou uniknout informace o plaintext zařízení, které by jinak zůstaly skryty.

Příkladem může být typ souborového systému (uvolněné místo vytváří vzor specifický pro typ souborového systému) [6].

Před povolením TRIM přes šifrovaný disk je třeba velmi důsledně zvážit, zda to v konkrétním případě nepředstavuje reálnou hrozbu.

7.3 Použití TPM

TPM (Trusted Platform Module) je speciální kryptografický čip, osazovaný přímo na základní desku počítače, který by měl sloužit jednak jako bezpečné úložiště pro klíče, ale zejména pro zajištění bezpečného startu systému (Trusted Boot).

Problém je, že kompletní a funkční implementaci bezpečného bootu nikdo zatím spolehlivě neposkytuje, nemá smysl tedy rozebírat tento případ.

Pokud se TPM používá, pak většinou jen pro uložení klíčů. (A i na tento účel TPM obsahuje pouze velmi omezenou paměť.)

Hlavním problémem při použití šifrování, které provádí systémový procesor, však je, že klíč musí opustit TPM a být přítomen v paměti, nebo alespoň v registru procesoru. V jistých implementacích je klíč přítomen v paměti i před autentizací uživatele.

Současné použití TPM je spíše technickou zajímavostí a marketingovým trikem, než reálnou možností jak zvýšit bezpečnost.

7.4 Šifrování disku vs šifrování síťového připojení disku

NAS (Network Attached Storage) je již z názvu disk připojený přes počítačovou síť. Příklad síťového připojení disku je iSCSI – vzdálený SCSI disk, kde diskové operace se posílají po síti a provedou na vzdáleném stroji.

Pokud exportované zařízení je šifrované, neznamená to zároveň, že i komunikace je bezpečně šifrována.

Šifrovaný disk

- obecně neplatí, že útočník má k dispozici historii sektorů (ukradený disk obsahuje jen poslední data)
- použití blokového módu je specifické pro sektor (IV pro sektor se nikdy nemění)

U **síťové komunikace** je předpoklad opačný

- útočník může lehce zaznamenávat síťový provoz, tedy má k dispozici historii
- IV či nonce (jednorázová hodnota) se při šifrované komunikaci nesmí opakovat (jinak by útočník mohl vložit zprávu případně zopakovat komunikaci)

Nelze tedy tyto implementace šifrování zaměňovat.

Nejjistější cestou je obě situace řešit nezávisle, tedy použít šifrování disku jako ochranu před únikem dat při ztrátě média a šifrovat komunikaci mezi serverem a klienty standardním síťovým protokolem.

7.5 Chyby média a důsledky

Chyba (např. změna bitu v RAM) je při dešifrování propagována, a to minimálně do jednoho celého bloku. Jinými slovy, to, co na normálním sektoru bude jednobitová chyba, při dešifrování se projeví v porušení mnohem většího bloku dat.

To je důvod, proč chyba média (RAM, disk apod.) má při použití šifrování vetšinou horší a viditelnější následky.

8 ... a zajímavé útoky

Pokud má útočník opakovaný přístup k hardware, případně má přístup do systému s administrátorskými právy, nelze se v principu úniku šifrovacího klíče a dat bránit nijak. Do této kategorie spadají všechny rootkity, viry, ale i některé hw útoky.

Typickým zástupcem této kategorie je *Evil Maid*: Počítač ponechaný na hotelovém pokoji pokojská modifikuje nešifrovaný zavaděč systému, tak, že například uloží zadávané heslo. Jakmile se uživatel přihlásí, vrátí se zpět, heslo i s kopií disku si odnese a uvede disk do původního stavu.

Stále platí heslo: **Počítač ponechaný bez dohledu již nemusí patřit jen vám.**

8.1 Útoky na algoritmus

Symetrické blokové algoritmy (jakým je AES) již poměrně dlouho odolávají intenzívní analýze, rozhodně je nelze tedy považovat za nejslabší článek.

Útok hrubou silou je v praxi, při daných délkách klíče, nerealizovatelný. To ovšem platí *jen pokud je klíč náhodný* – pokud se ukáže, ze generátor náhodných čísel byl defektní, útok hrubou silou může být realizovatelný.

Útok na nevhodně zvolený mód a IV je již realizovatelný i v praxi. Zástupcem je *watermarking* při použití CBC módu a predikovatelného IV. Tento útok lze snadno eliminovat vhodným použitím funkce pro výpočet IV, přesto se stále objevují systémy, které tuto zranitelnost obsahují.

Další skupina útoků je postavena na *využití nějakého vedlejšího kanálu*, spojeného s konkrétní implementací. Příkladem takového útoku je měření doby operace a manipulací s cache procesoru (AES cache collision attack).

8.2 HW útoky

Při použití standardního hardware je třeba se s určitými vlastnostmi smířit a zamezit jejich zneužití jinými způsoby.

HW keylogger

Prvním příkladem útoku na HW je použití *hardwarového keyloggeru*. To je zařízení, které se transparentně připojí před vstup z klávesnice a zaznamenává všechny stisknuté klávesy.

Zařízení může být implantováno přímo do čipu v klávesnice či do notebooku, z laického pohledu nelze jednoduše přítomnost takového zařízení detekovat.

V určitých případech se lze bránit použitím tokenu, který sám simuluje klávesnici a heslo odešle mimo odposlouchávaný kanál.

Cold boot

Protože šifrování provádí procesor, klíč musí být uložen v paměti.

Zástupce útoku na získání klíče z paměti je **Cold boot** [8] ať již ve variantě podchlazení paměťového čipu a přenesení do jiném systému (nízká teplota zajistí, že obsah paměti zůstane po jistou dobu přítomen i bez napájení) nebo jen pomocí kopie paměti po násilném resetu.

Vzhledem k popularitě tohoto útoku se objevují implementace, které se snaží problém vyřešit tak, že *klíč neopustí registry procesoru* (TRESOR, Loop Amnesia, Frozen cache).

Praktické použití těchto implementací je velmi sporné. Ani jedno z nich neřeší virtualizaci ani problém, co dělat v situaci, kdy nějaký kód registry procesoru uloží do paměti pomocí nějakého triku.

Zato dopad na výkonnost je výrazný, případně je použití omezeno jen na jedno mapované zařízení. Všechna tato řešení navíc vyžadují specifické vlastnosti procesoru, které nemusí být dostupné na všech architekturách.

Z pragmatického pohledu je lepší se smířit s omezeními a nastavit bezpečnostní politiku tak, že *počítač bez dozoru nikdy neobsahuje klíč v paměti.*

luksSuspend

Například LUKS poskytuje příkaz *luksSuspend*, který umožňuje zmrazit mapování a vymazat všechny klíče z paměti bez nutnosti odpojení souborového systému.

Příkaz *suspend* využívá obecné vlastnosti device-mapper zařízení, která umožňuje pozastavit IO operace (původně je určena pro změnu mapovací tabulky zařízení). LuksSuspend provede suspend dm-crypt zařízení a speciální zprávou oznámí, že se má vymazat klíč. To se provede nastavením všech aktivně

používaných kryptografických objektů na prázdný klíč (a souvisejících interních objektů).

Zpětná aktivace probíhá tak, že se nejdříve klíč dešifruje z LUKS (stejným způsobem, jako při aktivaci – uživatel musí zadat heslo) a po jeho nastavení v dm-cryptu se provede *resume*.

Přesto, že paměť stále může obsahovat některé sektory v otevřeném formátu, po luksSuspend neobsahuje vlastní šifrovací klíč (ani v jeho derivovaných podobách).

Literatura

[1] http://code.google.com/p/cryptsetup/

[2] http://loop-aes.sourceforge.net/

[3] http://www.truecrypt.org/

[4] http://windows.microsoft.com/en-US/windows7/products/features/bitlocker

[5] http://en.wikipedia.org/wiki/Comparison_of_disk_encryption_software

[6] http://asalor.blogspot.com/2011/08/trim-dm-crypt-problems.html

[7] http://code.google.com/p/cryptsetup/wiki/DMCrypt

[8] http://citp.princeton.edu/memory/

PKI, sen nebo noční můra

Luděk Smolík

E-mail: lsmolik@web.de

Abstrakt

Pojem „Public Key Distribution System“ (dnes pod názvem Public Key Infrastructure PKI) se zrodil v květnu 1975 a byl poprvé zmíněn v pověstné publikaci autorů Whitfiled Diffie a Martin Hellman: New Directions in Crypthopraphy v roce 1976. S odstupem téměř jedné generace je zajímavé, ale i poučné, rekapitulovat, jakým směrem se PKI koncept dodnes opravdu vyvinul. PKI byla nejdříve představená jako základní bezpečnostní platforma bouřlivě se vyvíjející informační technologie, ve které se nezdálo být technicky nic nemožné. Leč jak ani stromy do nebes nerostou, tak se v průběhu posledních deseti let ukazuje, že praxe a trh upřednostňují PKI řešení, která jsou ve srovnáni s původním univerzálním „všeumělem“ daleko střízlivější. Samozřejmostí se dnes staly především automatizované „na míru šité“ PKI aplikace v čistě digitálním světě bez interakce uživatele. Naproti tomu troskotá vize osobního elektronického podpisu digitálních dokumentů na banalitách a věčné slabině, jako je zapomenutí 6 místného pinu na všeobecné uživatelské informovanosti, ale i na otázce dlouhodobé bezpečnosti matematických algoritmů, a v neposlední řadě i na rozdílech nacionálních zákonů. Zajímavý a téměř neuvěřitelný je i vývoj staronové konkurence dynamicky digitalizovaného vlastnoručního podpisu, který se dnes řadí do biometrické autentizace podobně jako otisk prstu.

Public Key Cryptography a Infrastructure a její problémy v kostce

Když Whitfield Diffie a Martin Hellman v roce 1976 představili průkopnický koncept kryptografie veřejných klíčů (PKI) jako globální řešení bezpečné komunikace, navrhli pragmaticky zcela jednoduše upravený telefonní seznam, ve kterém se daly vyhledat veřejné klíče. Místo adresy a telefonního čísla měl být

Kdo najde v článku pravopisné chyby, může si je nechat.

Obr. 1: Ralph Merkle, Martin Hellman, Whitfield Diffie (1977)

ke jménu přiřazen veřejný klíč. Pokud byste chtěli najít veřejný klíč pana Jana Nováka, vyhledáte si ho v adresáři..., ale pozor, kolik takových Nováků na světě je... nemluvě o počtu Mohammedů a Changů?

V přehledném telefonním seznamu Stanford Computer Science deparmentu to tenkrát fungovalo. Ve větším měřítku třeba města New York nebo celého státu je to již otázkou a o celosvětovém nasazení v Internetu ani nemluvě. Tady začaly ty dětské nemoci, o kterých bude řeč.

Algoritmus Diffie-Hellman-Merkle pro výměnu kryptografických klíčů sice poskytl implementaci pro bezpečnou distribuci, ale neimplementoval digitální (elektronický) podpis. Tři výzkumníci MIT (Massachusetts Institute of Technology), Ronald Rivest, Adi Shamir, and Leonard Adleman (RSA) motivovaní lekturou práce Diffie, Hellman a Merkle počali bádat po praktické matematické funkci pro implementaci do kompletní Public Key Cryptography (PKC). Kandidátů měli 40 a nakonec objevili elegantní algoritmus dnes známý pod zkratkou RSA, který přesně pasoval k požadavkům PKC. V srpnu 1977 publikoval americký magazín *Scientific American* v čtivé rubrice *Mathematical Games* od Martina Gardner tento průlom v kryptografii. V očích NSA vznikl nový druh nebezpečí ohrožující národní bezpečnost a NSA požadovala ukončení distribuce výsledků, což vedlo tak svobodné duchy svého času, jako byli Rivest, Shamir a Adleman, k reakci a publikaci podrobného popisu jejich algoritmu v únoru 1978 v magazínu *Communications of the ACM*.

Až v roce 1997 se objevily informace, že již v letech před 1970 se v Britském Království v tak tajemných organizacích jako je Government Communications Headquarters (GCHQ) eventuálně podobná otázka tajně studovala. Britský matematik, zaměstnanec GCHQ, James Henry Ellis byl inspirován publikací[1] Bell

[1]Pro zainteresovaného čtenáře: technické řešení využití eletronického šumu, tak jak zřejmě popisovala ztracená publikace Bell Laboratories z roku 1944 je například publikovaná v [12].

Laboratories z roku 1944 o možnosti bezpečné komunikace dvou účastníků bez předcházejících kryptografických příprav (tenkrát se jednalo o komunikaci pomocí telefonního spojení a využití elektronického šumu). Tato publikace z Bell Laboratories se sice nezachovala, ale Ellis ji ve své publikaci z roku 1970 [9] zmiňuje a vede formální existenční důkaz matematického řešení pro utajení komunikace bez předcházející výměny tajných klíčů a nazývá ji Non-Secret Encryption (NSE).

Ve své pozdější publikaci: *The history of Non-Secret Encryption* [10], popisuje Ellis vznik PKI z jeho pohledu. Pro zajímavost zde cituji několik úvodních odstavců z této publikace:

Public-key cryptography (PKC) has been the subject of much discussion in the open literature since.

Diffie and Hellman suggested the possibility in their paper of April 1976 (reference 1). It has captured public imagination, and has been analysed and developed for practical use. Over the past decade there has been considerable academic activity in this field with many different schemes being proposed and, sometimes, analysed.

Cryptography is a most unusual science. Most professional scientists aim to be the first to publish their work, because it is through dissemination that the work realises its value. In contrast, the fullest value of cryptography is realised by minimising the information available to potential adversaries. Thus professional cryptographers normally work in closed communities to provide sufficient professional interaction to ensure quality while maintaining secrecy from outsiders. Revelation of these secrets is normally only sanctioned in the interests of historical accuracy after it has been demonstrated clearly that no further benefit can be obtained from continued secrecy.

In keeping with this tradition it is now appropriate to tell the story of the invention and development within CESG[2] *of non-secret encryption (NSE) which was our original name for what is now called PKC. The task of writing this paper has devolved on me because NSE was my idea and I can therefore describe these early developments from personal experience. No techniques not already public knowledge, or specific applications of NSE will be mentioned. Neither shall I venture into evaluation. This is a simple, personal account of the salient features, with only the absolute minimum of mathematics.*

The story begins in the 60's. The management of vast quantities of key material needed for secure communication was a headache for the armed forces. It was obvious to everyone, including me, that no secure communication was possible without secret key, some other secret knowledge, or at least some way in which the recipient was in a different position from an interceptor. After all, if they were in identical situations how could one possibly be able to receive what

[2]CESG, Communications-Electronics Security Group, pracovní skupina v GCHQ.

the other could not? Thus there was no incentive to look for something so clearly impossible.

The event which changed this view was the discovery of a wartime, Bell-Telephone report by an unknown author describing an ingenious idea for secure telephone speech (reference 2). It proposed that the recipient should mask the sender's speech by adding noise to the line. He could subtract the noise afterwards since he had added it and therefore knew what it was. The obvious practical disadvantages of this system prevented it being actually used, but it has some interesting characteristics. One of these, irrelevant to the main theme, is the amusing party trick of using the negative of the speech signal as the added noise. This leaves no signal on the line but the received signal unimpaired. This is easy to do and somewhat startling, but a simple analysis of the feedback shows that it is simply an amplifier with a low input impedance which shorts out the line.

...

První matematická formulace NSE byla publikována 1973 opět v tajném dokumentu CESG autorem Clifford Cocks [11]. Tato práce popisuje algoritmus, který je RSA algoritmu podobný a liší se v tom, že šifrovaní a luštění nejsou tak elegantně inverzní, jako je tomu u RSA. A pro úplnost, v roce 1974 publikoval Malcolm Williamson [13] metodu pro výměnu klíčů, která je podobná metodě Diffie, Hellman a Merkle publikované veřejně až několik let později.

Zcela původní počátky elegantní myšlenky asymetrické šifry jsou zajisté ještě starší a vyrostly z numerické matematiky, která se zabývá faktorizací přirozených čísel již od Antiky. Možná jedna z úplně prvních zmínek v literatuře se najde v knize britského ekonoma William Stanley Jevons (1835–1882) *Principles of Science*, na straně 123: *„Can the reader say what two numbers multiplied together will produce the number 8616460799? I think it unlikely that anyone but myself will ever know."* Výrok pana Jevons se dnes jeví jako úsměvný a faktorizace tak jednoduchého produktu je již známa mnoho let ale dodnes není prokazatelné, je-li faktorizace principiálně tak těžká a nebo nejsou-li efektivnější metody zatím jen neznámé.

Ale proč o tajné minulosti tak široce psát? Uvedený příklad svobodného amerického ducha na jedné straně a zatajení vědeckeckého pokroku v jiných zemích na druhé straně, nás musí varovat. Co je dnes veřejně publikováno, nemusí být vše, co je dnes neveřejně známo. Pro PKI a budoucí provozovatele může nastat dilema, komu vlastně důvěřovat.

Řada států na celém světě připravila se zákony o elektronickém podpisu na přelomu století legislativní půdu pro elektronické využití asymetrického šifrování. Se zavedením infrastruktury veřejných klíčů se tak vytvořily základy pro využití autentizace, identifikace a elektronického podpisu v obchodních procesech. Leč opravdu globální nasazení asymetrické kryptografické technologie s upotřebením kupříkladu čipových karet nebo jiných tokenů dnes stále ještě kulhá. Pro přístup k aplikacím se i dnes používá prvořadě klasická kombinace hesla a uživatelského

jména. Silná identifikace a autentizace s využitím kryptografie se instaluje jen u vykázaných aplikací ve specifickém úzkém kruhu. Zajímavý aspekt je, že se o neúspěchu PKI v odborné literatuře téměř nic nepublikuje a problém se oficiálně jen zřídka analyzuje.

Jaké jsou tedy hlavní důvody, proč se PKI nevyvíjí tak, jak si to experti představovali?

Silnice, most, železniční trasa, elektrické vedení a všechny další příklady reálných infrastruktur mají něco společného, jsou naplánovány tak, že přináší většinou trvalou amortizaci. U vysoce virtuálních infrastruktur, jako je PKI, jednak chybí znalosti, jak takovou amortizaci spočítat a dále stále ještě ovlivňuje prudký vývoj techniky, algoritmů, protokolů, norem a všeho kolem – nepřihlíženje na počítačovou vyspělost uživatele – životnost PKI. Životnost a trvanlivost produktu jsou ale základní předpoklady pro ekonomický úspěch. Negativním příkladem poslední doby je právě prasklá Dotcom bublina v roce 2000.

Obr. 2: Příklad úspěšného spojení infrastruktury a produktu (vlevo infrastruktura, vpravo produkt)

Předpoklad pro správně plánovanou, dlouhodobou životnost PKI je zodpovědnost za trvanlivost použitých algoritmů a kryptografických klíčů. Především pro státní aplikace (všechny druhy legitimací, archivované dokumenty, závazná komunikace s úřady atd.) hraje dlouhodobá životnost naprosto klíčovou roli. Zodpovědnost není levná věc a zároveň předpokládá i vysokou vyspělost celého technologického řetězce. Pro nasazení PKI existují dva finančně motivované argumenty: jednak inicializuje PKI proces digitalizace prakticky všech komunikačních kanálů a přispívá tak ke snížení finančních nákladů a za druhé se digitalizované procesy snadněji automatizují, sjednocují a urychlují. Problémem zůstává, že mnohdy ani zodpovědní manažéři a personál nemají přehled o finančních nákladech respektivně výpočet nákladů je často složitý a nepřehledný. I samotný zisk z instalované PKI a riziko provozu jsou obtížně kvantifikovatelné. Z těchto důvodů je výpočet úspor v reálných finančních hodnotách neobjektivní.

Na druhé straně je ale pro funkci PKI jako iniciátora nových postupů právě tento důkaz hospodárnosti a užitku na volném trhu rozhodující.

Standardní ekonomické indikátory, jako jsou kupříkladu ROI (Return On Invest), ROSI (Return On Strategic = Security Invest) a nebo NPV (Net Present Value), často ukazují negativní výsledek a mluví proti investici do PKI, i když se zdá být taková investice logicky rozumná.

Dalším problémem je, že nositelé nákladů na vznik PKI nemají často žádný přímý prospěch z investic. Na úrovni infrastruktury samotné nevzniká žádný zjevný transfer mezi náklady a užitkem, který by se dal zpoplatnit. Viditelný užitek vzniká až na úrovni procesů a transfer poplatků je na trhu doposavad težce prosaditelný.

Dále je třeba poznamenat, že PKI sama není specifická aplikace a z bezpečnostního hlediska nepřináší žádnou hodnotu. Jak již slovíčko infrastruktura v sobě nese, je to pouze platforma pro něco dalšího, přidaného.

Současné instalace PKI-technologie nejsou zatím pro denní potřebu tak užitečné, jak by bylo potřeba.

Především stav interoperability mezi různými PKI a jejich aplikacemi není ani po tolika letech od prvních publikací a po vypracováni mnoha nových konceptů pro uživatele příznivý. Ba dalo by se říci, že se koncept PKI vyvíjí nakonec úplně jiným směrem. Toto platí hlavně pro doslova klíčový proces, totiž správu a použití kryptografických klíčů.

Akceptace PKI řešení se dá navýšit především zjednodušením uživatelského rozhraní. Dále se úkazalo v praxi, že nároky na uživatelskou podporu jsou vyšší než u jiných aplikací, což znamená, že banální opatření, jako je například kvalita uživatelské podpory třeba v call centrech, nesmí být kompromitována nekvalifikovanými zaměstnanci.

Tržní jeviště pro PKI aplikace se dá rozdělit na tři scény, které mají každá svoji dynamiku:

- Hromadný trh kupříkladu Home-Banking a Online-Shopping. Zde stojí požadavky na jednoduchost a transparenci v popředí a tím brání de facto komplexní technologii jako je PKI.

- V oblasti jednotlivých firem a koncernů je flexibilita nejdůležitější kritérium. Zde existuje převážná většina dnešních specifických a ohraničených PKI řešení bez nároku na standardizaci a na stupeň bezpečnosti.

- ‚Par excellence' je nasazení PKI pro ochranu státních zájmů, kupříkladu pro elektronické občanské průkazy a pasy. Zde je dán bezpodmínečný požadavek na nejvyšší stupeň bezpečnosti, standardizaci, flexibilitu při výměně kryptografických algoritmů, nasazení budoucích biometrických metod apod.

Jaké jsou technické požadavky aneb pro pivo se džbánem až se ucho. . . ?

Závažnost bezpečnostních IT-technologií roste především v důsledku online podnikání v B2C a B2B sektoru. Nové právní předpisy jako Sarbanes-Oxley-Act nebo Basel II požadují organizační, eficientní a proaktivní nasazení IT-bezpečnosti. [2]

Většina zemí se zavedeným zákonem o elektronickém podpisu zveřejňuje pravidelně doporučení pro použití kryptografických algoritmů, hashovacích funkcí, metod kvalifikovaného elektronického podpisu a generování náhodných čísel (např. [3, 4, 5]). Jedno z doporučení je i záruka pro bezpečnost platnosti kvalifikovaného elektronického podpisu s výhledem na nejméně n let. N je číslo řádově 5 až 10 let. Toto doporučení je kompromis mezi *přáním* a *realitou* a je pro praktické a ekonomické nasazení s amortizací investic s daleko delším časovým horizontem nedostačující.

Jako příklad zde uvádím současná doporučení do roku 2017 a krátký souhrn parametrů pro použití kryptografických prostředků z německých katalogů algoritmů. Katalogy obsahují podrobnější matematický popis těchto parametrů [3]:

Hashovací funkce

SHA-1 a RIPEMD-160 jsou povoleny do konce roku 2015 jen pro kontrolu kvalifikovaného certifikátu,

SHA-224 pro tvorbu a kontrolu kvalifikovaného elektronického certifikatu do konce 2015,

SHA-256, SHA-384 a SHA-512 pro tvorbu a kontrolu kvalifikovaného elektronického certifikátu do 2017.

Algoritmy pro elektronický podpis

RSA-1976 bitů nejméně, RSA-2048 bitů doporučených do konce roku 2017,
DSA-2048 a $q = 224$ bitů do konce roku 2015, doporučení von DSA-2048 a $q = 256$ bitů do konce roku 2017,

Podobná jsou doporučení pro DSA variantu spočívající na eliptických křivkách $E(F_p)$ a $E(F_{2m})$.

Generátory náhodných čísel

Jako zdroj klíčového materiálu se důrazně doporučuje fyzikální generátor náhodných čísel.

Tam, kde není fyzikální generátor dostupný, se dá alternativně použít deterministický generátor s častou obnovou počátečního náhodného semínka. Kupříkladu do konce roku 2017 se doporučuje délka semínka 120 náhodných bitů.

Kontrola kvality generovaných náhodných čísel musí probíhat v obou případech pomocí matematických statistických testů online. Zároveň se v ideálním případu generovaná náhodná čísla nikde nehromadí a nezapisují. Tyto poslední dva požadavky jsou k sobě ortogonální a prakticky se vylučují. Právě zde leží hrozba pro budoucnost, zjistí-li se napaditelné nedostatky v minulosti vytvořených pseudonáhodných čísel.

Katalog algoritmů [3], jak ho vydáva kupř. německá spolková agentura, je základní podmínkou pro funkčnost spolehlivých systémů po delší časové období a je předpokladem pro zavedení nejen standardních postupů na národní úrovni ale i pro globální interoperabilitu. Na mezinárodní úrovni jsou cenné doporučení podle „Suite B“ od NSA [4].

Tento katalog doporučuje např aktuálně :

Encryption

AES-128 a AES-256 (Advanced Encryption Standard) [5]

Digitale Signatures

ECDSA-256 a ECDSA-384 (Elliptic-Curve Digital Signature Algorithm) [6]

Key Agreement

EC DH (Elliptic Curve Diffie-Hellman) nebo EC MQV (Menezes-Qu-Vanstone) s NIST P-256 bzw. NIST P-384 [7]

Hash-Function

SHA-256 a SHA-384, až SHA-512 (Secure Hash Algorithm) [8]

Zde je nutné poznamenat, že bezpečnost dnes nasazených kryptografických metod je dána kombinací výkonnosti výpočetní techniky a složitostí použitého matematického problému. Bezpečnost kryptografie tedy spočívá a vždy spočívala v tom, že ještě nikdo lepší matematický postup nenašel. Pokroky na poli numerické matematiky a v poslední době i v zcela jiných přírodovědných odvětví jsou pro posouzení bezpečnosti důležitými měřítky, i když bez vizionářských ambic jen těžce odhadnutelné.

Dnes je RSA algoritmus jakož standard asymetrické kryptografie široce implementovaný. Jeho bezpečnost je postavena na dostatečně složitém matematickém problému faktorizace velkých čísel.[3]

[3]Podobné platí kupříkladu pro eliptické křivky a výpočet diskrétních logaritmů.

Ohrožení bezpečnosti v důsledku pokroku výpočetní techniky a algoritmů se řeší pragmatickým zvyšováním délky kryptografického klíče. Aktuální vývoj na poli kvantových počítačů je hrozba, které se tímto způsobem čelit nedá. To stejné platí i pro vývoj algoritmů ale i fundamentálního poznání matematiky. Kvantový počítač, bude-li jednoho dne fungovat, by mohl řešit faktorizaci velkých čísel a všeobecně všechny deterministické problémy prakticky hrubou silou v polynomiálním času. K tomu potřebný algoritmus pro faktorizaci publikoval Peter. W. Shor jiz v roce 1994 [14]. Technické řešení spočívá v tak zvané kvantové Fouriertransformaci. Hned po zveřejnění algoritmu se ale vrhli fyzici i na vývoj klasických zařízení, která by mohla používat Shorův algoritmus bez kvantové fyziky. Příkladem je využití NMR (Nuclear Magnetic Resonance), ultra studených atomů a dalších.

I výzkum na poli DNA-Computing je jako hrozba relevantní. Bylo dokázáno v [15, 16], že i použití masivní paralizace výpočtů v molekulárních počítačích je vhodné pro zlomení kryptografického klíče.

Obrana proti těmto hrozbám by mohla být:

- vývoj alternativních kryptosystémů, například Lattice-Based Cryptosystems,
- narůst délky kryptografických klíčů v současných algoritmech.

Jako vždy je úřednický přístup k řešení podobných problémů pragmatický, a proto se dnes upřednostňuje druhé řešení. Z tohoto hlediska se dnes bezpečnost kryptografických algoritmů postavených na RSA a na eliptických křivkách hodnotí kladně alespoň na dalších 10 let.

Ale i budoucí řešení fundamentálních problémů matematiky jsou stálou a nekalkulovatelnou hrozbou pro RSA a všeobecně pro kryptografii. Připomeňme si matematickou Zeta-funkci zavedenou Eulerem, který jako první též ukázal její propojení k prvočíslům. Matematik Bernhard Riemann (1826–1866) později rozvinul Eulerovu Zeta-funkci v komplexní rovině a vytvořil možná nevědomky Eldorádo pro studium prvočísel. Riemannova komplexní Zeta-funkce obsahuje v sobě veškeré vlastnosti prvočísel, známé a i ty ještě neznámé. Příkladem je proslulá, dodnes nedokázaná Riemannova domněnka z roku 1859, v které je formulována hypotéza o rozložení netriviálních kořenů[4] komplexní Zeta-funkce. Důkaz Riemannovi domněnky by umožnil přístup k výpočtu rozložení prvočísel, to znamená analytickou formuli pro přesný počet prvočísel menších než nějaká hranice x. To by znamelo prakticky okamžité prolomení asymetrické kryptografie.

[4] Jak reálná tak i imaginární část Zeta-funkce má zde nulovou hodnotu. Všechny triviální kořeny se nacházejí na reálné ose. Riemannova domněnka tvrdí, že se všechny netriviální kořeny nacházejí na přímce paralelní k imaginární ose a procházející reálnou osou při 1/2.

Aktuálně se zdá, že hashovací funkce mají daleko kratší životnost ve srovnání s šifrovacími algoritmy.

Kolize s hashovací funkcí MD4 byly nalezeny již 6 let po zavedení [17]. Jelikož jsou funkce SHA-0 a SHA-1 postaveny na podobném algoritmu jako MD4, jsou tyto teoreticky též prolomeny. Potud způsobují hashovací funkce současně největší problémy a řešení musí zaručovat dlouhodobější bezpečnost.

Pro typická uplatnění PKI na nejvyšší bezpečnostní úrovni, jako je kupříkladu elektronická zdravotní karta, elektronické pasy apod., se předpokládá bezpečnost použité metody na příštích 30 let. Tyto předpoklady dnešní metody ještě zdaleka nesplňují. Ba dokonce i otázka, co bude za 20 let se nedá snadno odpovědět. Jediným současným schůdným řešením je dostatečná zásoba alternativních algoritmů a jejich lehká výměna v aplikacích. Zde se jeví četná různorodost implementace stejného algoritmů jako největší problém. Řešení je zavedení společné kryptografické API, podobně jak je tomu u Java Cryptographic Architecture (JCA) nebo Microsoft Crypto API. Stejně snadno se musí nechat měnit certifikáty a klíče. Zajímavá je v tomto kontextu propracovaná knihovna http://www.FlexiProvider.de, která vedle standardních algoritmů obsahuje i několik algoritmů, které by měly čelit útokům s kvantovými počítači (na stránce http://FlexiProvider.de pod *PostQuantumProvider*).

IEEE P1363 (Institute of Electrical and Electronics Engineers) je tradiční projektová skupina, která pracuje na specifikaci a standardizaci PKC. Těžištěm práce jsou jak standardní postupy, tak i vývoj nových, kupříkladu Lattice-Based Public-Key kryptografie, která by měla být též bezpečná před útoky s kvantovými počítači (http://en.wikipedia.org/wiki/NTRUEncrypt). NTRU je především podstatně rychlejší ve srovnání s RSA. NTRUEncrypt a NTRUSign jsou dnes již nasazeny v bezdrátových sítích.

Další vize budoucnosti, jak docílit dlouhodobou bezpečnost v kryptografii, je využití principů kvantové fyziky. Základní úvaha je postavena na možnosti důvěrného přenosu informací pomocí detekce jednotlivých elementárních částic v tak zvaném kvantovém kanálu [18, 19]. Při odposlechu kanálu se kvantový stav elementární částice měřením poruší a komunikační partneři mohou tyto změny statisticky odhalit. V tom okamžiku vědí, ze jejich komunikace není již dále bezpečná. Ale i zde existuje doposavad teoretické ohrožení klonováním původně jednotlivých elementárních částic.

Faktor nebezpečný uživatel

V souvislosti s bezpečností informační technologie a zavedením nových technologií se diskutuje další slabé místo: člověk = uživatel samotný. Typický uživatel, který obsluhuje systém nebo software, je experty jmenován jako nejslabší článek. Pravdou je, že jsou systémy často relativně bezpečné ale nejsou adekvátně

uzpůsobeny pro využití člověkem. Uživatelé zapomínají hesla, PINy, nebo si je zapisují a nedrží v tajnosti. Studie dokazují, že útočník nenapadá kryptografické metody, nýbrž hledá místo k útoku tam, kde se dostane nejsnáže k cíli a to jsou převážně implementace algoritmů a na druhé straně uživatel samotný. Doporučení k používaní silných to jest skoro nahodých hesel je kontraproduktivní, neboť úsporová funkce lidské paměti nepodporuje nelogické vzpomínky, člověk je prostě zapomene. Návody, jak si složité heslo přesto zapamatovat, jsou nakonec stejně jen další algoritmy, problém se zkomplikuje ale nevyřeší. Ovšem deklarovat uživatele jako slabinu, je hloupé a krátkozraké. Pro hromadné, akceptované nasazení je potřeba adaptovat technologii na člověka a ne obráceně. Využití biometrie je jedna z možností, toto dilema vyřešit. U klasické biometrie ale zatím stále přetrvává věčný problém s detekcí životnosti a jen spojení s novými metodami zde může přinést pokrok.

Ztroskotala PKI vize jednoduché asymetrické kryptografie?

Původní idea PKI byla formulována bezprostředně ve spojení se správou asymetrických klíčů. Předpokladem by byla hierarchická struktura v Internetu pro správu a spojení veřejného klíče s osobou, respektivně s jejím jménem a tím vznik jednoznačné elektronické identity uživatele.

Nikdy ale nebylo dostatečně vyřešeno, kdo má tuto infrastrukturu postavit, provozovat a jak s ní obchodovat. Proto je PKI myšlenka dnes i po tolika letech uskutečněná jen lokálně a parciálně a má mnoho různých řešení a specifických variant. Kritici by mohli vznést i argument: vždyť ono to nějak funguje i bez celosvětové PKI.

V dnešních variantách PKI chybí pro uživatele především transparence a možnost odhadu, co taková PKI přinese. Způsobilost a funkčnost technologie PKI pro bezpečnostní aplikace je uznávaná odborní- ky na celém světě. Nicméně očekávaný průlom PKI technologie v otevřeném Internetu dosud nenastal. Možná i právě tím, že Internet nemá hierarchickou strukturu. Důkazem je existence mnoha funkčních malých ale i větších koncernových PKI-řešení, ale vždy v uzavřeném kruhu uživatelů. Zde se dají veškeré strategické otázky bezpečnostní politiky ovládnout bez ohledu na složité domluvy se zbytkem světa.

Elektronický podpis občanů se měl stát vzorovou PKI aplikaci při styku s veřejnou správou Pro většinu uživatelů elektronického podpisu vznikají ale náklady bez viditelného přínosu. Naproti tomu vzniká znatelný přínos na straně druhé, u veřejných organizací. Dokud není vyřešen transfer nákladů, nebude tento druh masivní aplikace občany přijat.

Dalo by se to formulovat ještě všeobecněji: dokud nebude PKI levnější, to znamená ve světě Internetu prakticky bezplatná, nestane se součástí Cyberworld.

Literatura

[1] Diffie, w, Hellman, M.: New Directions in Cryptography, *IEEE Transactions on Information Theory*, Nov. 1976.

[2] Booker, R.: Re-engineering enterprise security. *Computers & Security* 25; 2006, s. mbox13–17.

[3] http://www.bundesnetzagentur.de/DE/Sachgebiete/QES/Veroeffentlichungen/Algorithmen/algorit hmen_node.html

[4] NSA, Suite B: http://www.nsa.gov/ia/programs/suiteb_cryptography/

[5] Federal Information Processing Standards Publication 197, November 26, 2001; *Announcing the Advanced Encryption Standard (AES)*. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[6] *Federal Information Processing Standards Publication 186-2*; Digital Signature Standard; Januar 2000. http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf

[7] *NIST Special Publication 800-56*, Recommendation on key establishment schemes; Januar 2003. http://csrc.nist.gov/CryptoToolkit/kms/keyschemes-Jan03.pdf

[8] NIST: *FIPS Publication 180-3: Secure Hash Standard (SHS)*, Oktober 2008.

[9] *The Possibility of Secure Non-Secret Digital Encryption*, J. H. Ellis, January 1970, http://cryptocellar.web.cern.ch/cryptocellar/cesg/possnse.pdf

[10] *The history of Non-Secret Encryption*, by J. H. Ellis. http://cryptocellar.web.cern.ch/cryptocellar/cesg/ellis.pdf

[11] *A Note on ‚Non-Secret Encryption'*, by C. C. Cocks. http://www.cesg.gov.uk/publications/media/notense.pdf

[12] *Totally Secure Classical Communication Utilizing Johnson (-like) Noise and Kirchoff's Law*, Laszlo B. Kish. http://arxiv.org/ftp/physics/papers/0509/0509136.pdf

[13] *Non-Secret Encryption Using a Finite Field*, by M. J. Williamson, 1974. http://www.cesg.gov.uk/publications/media/secenc.pdf

[14] *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, by Peter W. Shor. http://arxiv.org/PS_cache/quant-ph/pdf/9508/9508027v2.pdf

[15] Boneh, D., Dunworth, C., Lipton, R. J.: *Breaking DES Using a Molecular Computer*, Technical Report CS-TR-489-95, Princeton University, 1995.

[16] Boneh, D., Dunworth, C., Lipton, R. J., Sgall, J.: On the Computational Power of DNA, *Journal DAMTH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, Volume 71, 1996.

[17] Dobbertin, H.: *Cryptoanalysis of MD4. Proceedings of the 3rd Workshop on Fast Software Encryption.* Cambridge, UK; February 21–23 1996; Lecture Notes in Computer Science; Bd. 1039, s. 53–70, Berlin, Springer, 1996.

[18] Brassard, G., Crepeau, C.: 25 years of quantum cryptography. *SIGACT News* 27(3), 1996, s. 13–24.

[19] http://www.idquantique.com

Elektronické pasy v praxi

Zdeněk Říha

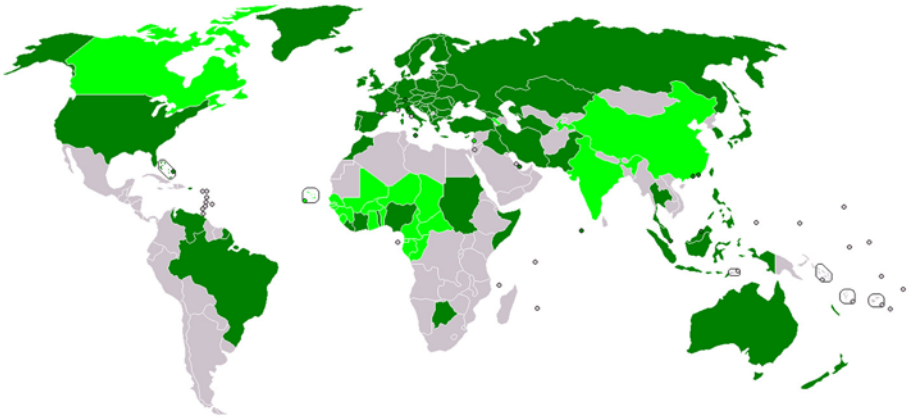
E-mail: zriha@fi.muni.cz

Elektronický pas obsahuje kromě klasické „knížečky“ bezkontaktní čipovou kartu s patřičnou anténou. Idea elektronického pasu je relativně stará. První zemí, která začala elektronické pasy vydávat, byla Malajsie a psal se rok 1998. Nicméně byly to až události z 11. září 2001, které vedly k většímu tlaku na bezpečnější doklady, kam elektronika v pase hezky zapadala. Celosvětový standard pro elektronické pasy byl vyvinut na půdě organizace ICAO. Šesté vydání dokumentu ICAO 9303 pro strojově čitelné dokumenty [5] bylo publikováno až v roce 2006, ale již před tím státy vydávající elektronické pasy používaly pracovní verze tohoto standardu a aktualizace dokumentu 9303 objasňující technické drobnosti se vydávají dodnes.

Na celosvětové úrovni je vydávání elektronických pasů dobrovolné, řada zemí však na výběr nemá. Země spadající pod americký VWP (Visa Waiwer Program) musí splňovat celou řadu podmínek na bezpečnost cestovních dokladů včetně elektroniky v pase. Země EU se pak zavázaly zavést povinně elektronické pasy (do roku 2006 s fotografií obličeje a od roku 2009 i s otiskem prstů) a s předstihem či zpožděním to také udělaly.

V dnešní době vydává elektronické pasy relativně velké množství států. Přesná čísla se získávají těžko, neboť zavádění elektroniky do pasu může být předmětem politických diskuzí (co bylo ohlášeno, včera nemusí dnes už platit) a v některých zemích jsou elektronické pasy vydávány jen některým skupinám držitelů (např. jen diplomatické pasy). Wikipedia [9] uvádí k červnu 2011 následující mapu zemí vydávajících elektronické pasy (tmavě zelená znamená pasy již dostupné držitelům, světle zelená znamená plány na budoucí zavedení). ICAO k červnu 2011 [6] uvádí 93 zemí vydávajících elektronické pasy (z toho 34 zemí ukládá pouze fotografie držitelů, 45 zemí ukládá fotografie držitelů a otisky prstů a 14 zemí ukládá pouze fotografie, ale plánuje ukládat i otisky prstů). Celkový počet již vydaných elektronických pasů se odhaduje na celkem 345 039 000. Počet zemí plánujících začít vydávat elektronické pasy během následujících 48 měsíců se odhaduje na 21.

Naštěstí (z hlediska držitelů) je obvykle dodržováno základní pravidlo, že staré neelektronické pasy neztrácejí svou platnost a elektronické pasy tedy zvyšují podíl v celkovém množství cestovních dokladů jen postupně.



Technologicky se vesměs jedná o stejná zařízení založená na standardu ICAO 9303 (tj. komunikace podle ISO 14443 rychlostí 106 až 848 kbs na vzdálenost 0–10 cm, práce s aplikací a soubory podle ISO 7816). I Malajsie začala v roce 2010 vydávat tyto pasy (jejich původní pasy se od standardu lehce lišily, neboť standard začal vznikat až v době, kdy Malajsie pasy již vydávala). Rozdílné je množství dat, volitelné bezpečnostní prvky a případné řízení přístupu k otiskům prstů. Výjimkou jsou Pákistán, Moldávie a Irák, které vydávají pasy nekompatibilní se standardy ICAO.

Digitální podpis

Důležitým povinným bezpečnostním prvkem všech elektronických pasů je digitální podpis uložených dat. Jedná se o běžný podpis využívající formát CMS (vycházející z PKCS#7) a struktura PKI není moc technicky náročná. Každá země zřizuje speciální certifikační autoritu (CSCA – country signing certification authority), která vydává certifikáty vydavatelům pasů v dané zemi (např. obce, ambasády; u nás je to Státní tiskárna cenin). Tito vydavatelé pasů (tzv. Document Signers – DS) pak digitálně podepisují data v pasech a přikládají k podpisům i svůj certifikát. Pochopitelně každý, kdo chce digitální podpis ověřit, musí mít k dispozici certifikát CSCA vydávající země. Něco jako kořenová CA zastřešující všechny CSCA světa a certifikující jejich certifikáty nebylo z hlediska celosvětové důvěry realizovatelné, takže pro získání CSCA certifikátů jednotlivých zemí je třeba využít diplomatické výměny certifikátů. To sice zní bezpečně, ale v praxi se tato výměna moc neosvědčila, neboť diplomaté nebývají moc technicky zdatní (a získaný certifikát se třeba nedostane v cílové zemi na správné místo) a rozdíl mezi zajištěním důvěrnosti a integrity neznají snad ani všichni

informatici... Jedné zemi bránila v zaslání certifikátu CSCA do zahraničí dokonce její místní legislativa.

CSCA mění svůj klíč/certifikát jednou za 3 až 5 let a důvěru v nový certifikát lze navázat důvěrou ve starý, jde tedy naštěstí obvykle jen o iniciální získání certifikátu. Klíče jednotlivých vydavatelů pasů v zemi se typicky často nekompromitují, přesto každá CSCA musí vydávat nejméně jednou za 90 dnů CRL. Specifikem v této oblasti je to, že i kdyby k odvolání klíče vydavatele pasu došlo, automaticky to neznamená neplatnost všech vydaných pasů.

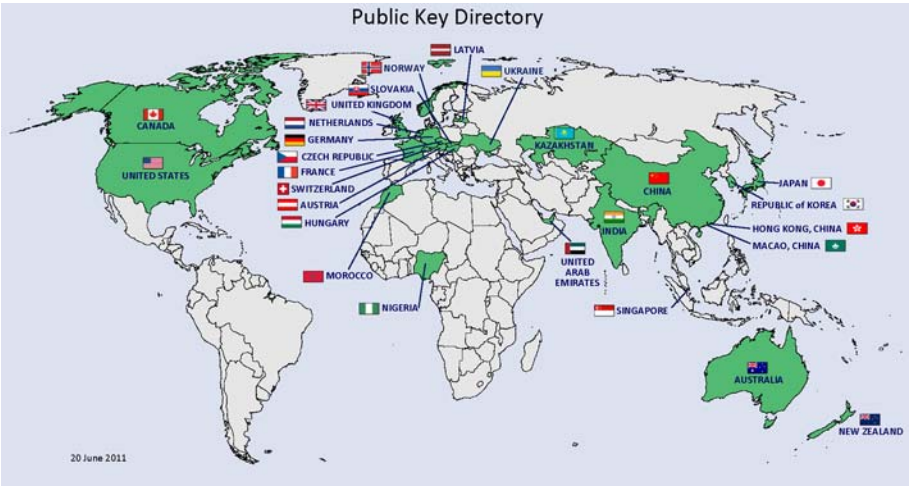
V oblasti správy klíčů vydavatelů pasů se čekala pomoc od ICAO PKD (Public Key Directory). Bohužel v získání CSCA certifikátů jiných zemí PKD nemůže pomoci přímo, a tak PKD nabízí pomoc ve dvou oblastech. Jednak je to seznam jednotlivých certifikátů vydavatelů pasů (DS) a jednak je to aktuální CRL. Dá se říci, že PKD tak vlastně řeší problém, který neexistuje, neboť certifikáty vydavatelů pasů (DS) jsou stejně uloženy v pase, s ověřením těchto certifikátů PKD nepomůže (je potřeba certifikát CSCA) a CRL se dá typicky stáhnout z webu. Navíc PKD obsahuje pouze data o zemích členů přihlášených do PKD a to jsou typicky právě ty země, kde sehnání aktuálního CRL je ten nejmenší problém. Přidaná hodnota PKD je tak relativně nízká, ale místo zrušení celého PKD se spíše hledaly cesty, jak PKD učinit užitečnějším. Novinkou jsou tzv. „master listy“, tj. seznamy certifikátů CSCA různých zemí podepsaných nějakou zemí, která tyto certifikáty důvěryhodně získala a používá je na svých hranicích. Stačí pak důvěra v tuto zemi a důvěryhodné získání certifikátu této země pro rychlé rozšíření seznamu certifikátů CSCA řady zemí světa. V současné době vydává takovéto „master listy“ [3] jen Německo (seznam obsahuje asi 45) a Austrálie (seznam obsahuje pouze 3 země). Další země však o vydávání takových seznamů uvažují.

Aktivní účast země v PKD (informace o certifikátech vydavatelů pasu a CRL dané země) není žádnou levnou záležitostí. Jednorázový poplatek při vstupu činí 56 000 USD a roční poplatek byl na rok 2011 stanoven na 43 000 USD. Stažení dat s PKD si může zkusit každý na adrese https://pkddownloadsg.icao.int, jedná se asi o 5 MB dat.

V červnu 2011 bylo členem PKD 27 zemí světa [4], včetně České a Slovenské republiky. Následující obrázek ilustruje členské země PKD.

Řízení přístupu k datům pasu

Rychlé čtení dat z e-pasů je nejjednodušeji proveditelné s nulovým řízením přístupu k datům. Ochrana soukromí držitelů pasů by pak spoléhala pouze na maximálně 10 cm čtecí vzdálenost, případně na stínění pasu v obalu či deskách pasu. Tyto původní ideje však rychle vzali za své. Ačkoliv pár zemí takové pasy opravdu vydávaly (např. Belgie), byly to spíše výjimky.



Z hlediska ICAO standardu **doporučeným**, z hlediska EU zemí pak **povinným** bezpečnostním prvkem se stalo tzv. základní řízení přístupu (anglicky basic access control – BAC). To vyžaduje autentizaci a ustaví klíč sezení, kterým šifruje další komunikaci. Použitý autentizační klíč je vytištěn v pase na datové stránce. Jedná se vlastně o část automatické čtecí zóny obsahující číslo dokumentu, datum narození držitele pasu a datum vypršení platnosti pasu. Ačkoliv tato data dosahují maximální entropie 56 bitů (a v případě alfanumerických „čísel“ pasů dokonce 73 bitů), v praxi je díky číslovacím pravidlům a jiným omezením entropie takto odvozených klíčů menší. Útok na nízkou entropii těchto autentizačních klíčů byl předveden pro případ nizozemských pasů, nicméně kromě velkého mediálního dopadu nevedl k žádným změnám.

Skupina odborníků vyvíjející patřičný ICAO standard nebyla nakloněna k jednoduché změně, která by při odvození klíče vzala v úvahu i jedno další pole s volitelným obsahem (které by tak mohlo hezky zvyšovat entropii autentizačního klíče) s tím, že by se narušila kompatibilita s již existujícími implementacemi. Jedinou možností pro zvýšení entropie klíče tedy bylo zavedení alfanumerických, zcela náhodných čísel pasů, což pro většinu zemí nebylo atraktivní z celé řady praktických hledisek. Jednou z mála výjimek bylo Německo, které zavedlo náhodná čísla pasů. Ačkoliv některé znaky čísla dokumentu jsou nenumerické, i tak není v použitém německém schématu dosaženo plné teoretické maximální entropie tohoto pole (některé prvky kódují vydávající spolkovou zemí apod.). Většina zemí se pak k problému stavěla tak, že zabezpečení dat je odpovídající jejich obsahu (ve srovnání s jinými možnostmi získání stejných dat).

Problém nízké bezpečnosti základního řízení přístupu však není možné ignorovat donekonečna. Zvláště pokud vezmeme v úvahu rostoucí rychlost počítačů

(a jiných zařízení schopných útočit na kryptografické klíče) a dlouhou platnost jednou vydaných dokladů (maximální doba platnosti je 10 let). Bylo tedy nutné najít řešení zahrnující nový, bezpečnější protokol a způsob realizace přechodového období [8].

Řešení existuje a jmenuje se dodatečné řízení přístupu [7] (anglicky Supplemental Access Control – SAC). Jedná se o další využití protokolu PACE, který je znám už ze specifikace německých občanských průkazů. PACE je zkratka z Password Authenticated Connection Establishment a jak již název napovídá, jedná se o ustavení (bezpečného) spojení s autentizací obou stran. To sice umí i základní řízení přístupu, základním rozdílem zde však je to, že bezpečnost PACE nezávisí na síle sdíleného tajemství (klíče, hesla atd.). I zde je totiž tím sdíleným tajemstvím několik položek z automatické čtecí zóny na datové stránce, ba dokonce jde o stejné tři položky se stejnou entropií.

Abychom se neztratili ve verzích jednotlivých protokolů, uvedu krátké shrnutí: Existují i pasy bez jakéhokoli řízení přístupu. O těch se dnes již moc nemluví a mohli bychom je nazývat pasy nulté generace. Pasy první generace jsou založeny na základním řízení přístupu (BAC). Pasy druhé generace také podporují základní řízení přístupu, ale obsahují dále otisky prstů, které jsou chráněny pomocí rozšířeného řízení přístupu (anglicky Extended Access Control – EAC). Rozšířené řízení přístupu zavádí autentizaci čipu (pro ověření autenticity dokladu) a autentizaci terminálu (pro řízení přístupu). U pasů se používá verze jedna rozšířeného řízení přístupu (přesněji je to EAC v 1.11). EAC se dočkalo i druhé verze (aktuálně jde o EAC v 2.05), to se však používá u německých občanských průkazů. EAC druhé verze používá také autentizaci čipu a terminálu (ale v obráceném pořadí než ve verzi 1) a zavádí protokol PACE. V EAC verze 2 jde o PACE verze 1. Ve specifikaci dodatečného řízení přístupu (SAC) se pak objevuje lehce modifikovaný protokol PACE, nazývaný jako PACE verze 2. Pasy podporující SAC se nazývají pasy třetí generace.

Detaily protokolu PACE najdeme v samotné specifikaci [7], analýzu bezpečnosti pak například v [1] nebo [2]. Protokol se skládá ze 4 základních kroků. Nejprve pas odešle náhodné číslo šifrované sdíleným tajemstvím čtecímu zařízení. Toto náhodné číslo je pak na obou stranách použito pro odvození náhodného generátoru grupy. Nad touto grupou s patřičným generátorem je pak spuštěn protokol Diffie-Hellman pro získání klíčů sezení. Nakonec se generují a na druhých stranách ověří autentizační tokeny (závisející na veřejném DH klíči druhé strany a klíči sezení). Existují varianty pro klasický protokol Diffie Hellman a jeho verzi založenou na eliptických křivkách.

Rozdíl mezi PACEv1 a PACEv2 spočívá ve způsobu, jak odvodit náhodný generátor grupy. Ve starší verzi 1 se používá německý přístup nazvaný obecné

mapování[1] (anglicky generic mapping), v novější verzi 2 se navíc může použít francouzský přístup nazvaný integrované mapovaní[2] (anglicky integrated mapping). Do problematiky zasáhlo i určité politikaření a fakt, že integrované mapování je patentováno. Na využití v elektronických pasech však byla poskytnuta bezplatná licence.

Pasy třetí generace využívající dodatečného řízení přístupu budou po dobu přechodové fáze (odhadované organizací ICAO na 10 až 20 let) podporovat jak nové SAC, tak i starší BAC, s tím, že čtecí zařízení na hranicích si protokol vybere. Pochopitelně aktualizované čtecí zařízení podporující SAC použije protokol SAC kdykoliv narazí na SAC pas. Nicméně pasy budou muset podporovat i BAC pro případ čtení v zařízeních nepodporujících SAC.

V zemích EU se očekává zavedení pasů třetí generace s SAC nejpozději od roku 2014.

Zavedení rozšířeného řízení přístupu

Nejpozději od června 2009 musí (skoro) všechny země EU vydávat pasy i s otisky prstů. Tyto pasy takzvané druhé generace jsou doplněny protokolem rozšířeného řízení přístupu pro zajištění řízení přístupu k otiskům prstů (jako více citlivým datům než k jiným údajům v čipu). Autentizace terminálu jako součást EAC má za úkol povolit přístup k otiskům prstů jen autorizovaným čtecím zařízením. Čtecí zařízení musí pasu předložit autorizační certifikáty a prokázat přístup k patřičnému soukromému klíči. Protože použité schéma nezná odvolání certifikátů, jsou certifikáty vydávány s krátkou dobou platnosti. Aby bylo možné číst otisky prstů z pasu země A v zemi B, je nutné mezi zeměmi A a B navázat vztah důvěry a pravidelně aktualizovat certifikáty. Iniciální navázání vztahu důvěry se neobejde bez osobnějších setkání, následné aktualizace certifikátů mohou vyžít navázání důvěry pomocí existujících klíčů.

Specifikace EAC původně předpokládaly emailovou aktualizaci certifikátů. Emailová komunikace není nijak speciálně zabezpečena. Vlastně se posílají běžné nezabezpečené emailové zprávy s domluveným předmětem emailu a přílohou obsahující certifikát, požadavek na vydání certifikátu apod. Protože emailová komunikace není nijak potvrzována je těžké zjistit, zda v případě nedoručení požadovaného certifikátu byl ztracen již požadavek nebo jen vytvořený certifikát.

[1] Obecné mapování používá pro eliptické křivky mapovací funkci $G' = s \cdot G + H$, kde s je pasem vytvořené náhodné číslo a H je bod získaný během anonymního ECDH protokolu. U celočíselného algoritmu DH se používá mapovací funkce $g' = g^s \cdot h$, kde s je pasem vytvořené náhodné číslo a h je číslo získané během anonymního DH protokolu.

[2] Integrované mapování používá pro eliptické křivky mapování $G' = f_G(R_p(s,t))$ a pro celá čísla mapování $g' = f_g(R_p(s,t))$, kde s a t jsou náhodná čísla generovaná pasem a čtecím systémem. R_p je pseudonáhodná funkce založená na použité blokové šifře v CBC režimu. Pro f_g a f_G je třeba nahlédnout do specifikace.

Navíc platnost klíčů je omezena a pokud certifikát není v době platnosti staršího klíče aktualizován, není již možné využít existující klíče pro získání nového certifikátu a je třeba zopakovat iniciální navázání důvěry.

Při pokusu o první implementace plného PKI se plně ukázaly nevýhody popsaného přístupu. Proto se některé aktivnější země EU dohodly, že kromě povinného emailového rozhraní vyvinou nový protokol založený na webových službách, který bude poskytovat podobnou funkcionalitu, ale spolehlivějším způsobem. Každá země vytvoří jediný kontaktní bod (anglicky Single Point of Contact – SPOC), který vyřizuje požadavky a zajišťuje notifikace. Komunikace je chráněna pomocí SSL/TLS. Z českého pohledu je zajímavé, že tato specifikace byla vydána jako ČSN 369791:2009.

Protože se rozhraní webových služeb osvědčilo, bylo dokonce na úrovni EU rozhodnuto, že toto rozhraní bude povinné místo původně zamýšleného emailu. V současné době probíhá výměna certifikátů mezi několika prvními zeměmi. Ani to však ještě neznamená, že se takto získané certifikáty jsou využívány na hranicích. A do úplné matice výměny certifikátů všech zúčastněných zemí je opravdu ještě daleko.

Využívání elektronických pasů na hranicích

Jedna věc je elektronické pasy vydávat a zcela jiná záležitost je pak elektronické pasy při ověřování totožnosti používat. Použití elektronické části elektronických pasů je možné rozdělit do dvou přístupů. V prvním případě se provádí ověření autenticity dokumentu a dat (čtení dat, ověření digitálního podpisu – pasivní autentizace dat, aktivní autentizace a autentizace čipu). V druhém případě se pak dále pokračuje biometrickým ztotožněním načtených biometrických dat s osobou předkládající cestovní doklad.

ICAO uvádí že v současné době celých 56 zemí začlenilo čtení ePasů do svých procesů na hranicích. Do této kategorie však spadá jak plné čtení a ověřování všech pasů, tak i jen první pokusy, testy apod. (například jen pouhé náhodné čtení vybraných pasů na několika vstupních bodech bez jakéhokoli ověřování). Zemí, které opravdu čtou elektronické pasy na svých hranicích je pouze 10 na celém světě. Jedná se o Austrálii, Kanadu, Německo, Indonésii, Japonsko, Nový Zéland, Portugalsko, Singapur, Spojené království a Spojené státy americké.

Diskuze o možném povinném čtení ePasů na hranicích v zemích EU či alespoň Schengenu zatím nevyústily v konkrétní legislativní kroky.

Co brání řadě zemí ve využívání ePasů na hranicích? Zřejmě nemalé náklady na úpravy systémů a relativně malá přidaná hodnota (zvláště pokud jde z hlediska pasažérů o neautomatizované systémy). Ačkoliv běžná RF čtečka schopná komunikovat s ePasem stojí kolem 1 000 korun, kompletní scanner pasu schopný rozpoznat automatickou čtecí zónu, zkontrolovat bezpečnostní prvky apod. vyjde řádově na 100 000 korun.

Bezpečný dokument obsahující biometrické údaje (kterým pas je) může být použit i pro automatizovanou kontrolu (průchod hranic apod.). V takovém případě se nejprve načtou data z pasu a ověří se (také se ověří autenticita dokumentu, možnost občana dané země využít automatizovaný systém atd.), teprve potom držitel vstoupí do nějakého boxu (či jinak vyznačeného prostoru) a je biometricky ztotožněn s daty v pase. Pokud ověření skončí úspěšně může cestující pokračovat dál, pokud neúspěšně, pak je na řadě běžná manuální kontrola.

Tyto automatizované systémy se nazývají ABC (z anglického Automated Border Control). Ačkoliv samotné odbavení není rychlejší než bezproblémová manuální hraniční kontrola, v jejich prospěch mluví především dobrá škálovatelnost (pokud to prostorové možnosti letiště dovolí). Dá se říci, že ABC systémy jsou dnes módou a žádné významnější letiště u toho nechce chybět. I v Praze se ABC systém plánoval, ale realizace se zřejmě z finančních důvodů odsouvá na neurčito.

ABC systémů je na světě celá řada, ty první z nich, ale nebyly založeny na elektronických pasech a vyžadovaly separátní bimetrickou registraci do daného systému. Ale až elektronické pasy pomohly odbourat tuto iniciální fázi, neboť digitálně podepsanou biometrickou šablonu si sebou přináší sám držitel pasu. Pionýry v této oblasti se staly letiště ve Faro (Portugalsko, systém RAPID) a v Brisbane (Austrálie, systém SmartGate). Později se podobné systémy objevily ve Spojeném Království (součást projektu e-Borders), Německu (EasyPass) atd. Až na výjimky jsou tyto ABC systémy založeny na rozpoznání obličejů neboť čitelnost otisků prstů z pasů chráněných rozšířeným řízením přístupu je v současné době problematická.

ABC systémy byly napadány pro jejich údajně nízkou bezpečnost [10]. Biometrická verifikace založená na rozpoznání obličeje bývá méně přesná než biometrické systémy založené na duhovkách či otiscích prstů. Navíc bezpečnost těchto systémů stojí a padá na ověření digitálního podpisu dat a ověření autenticity dokumentu. Iniciální problematická dostupnost CSCA certifikátů a snaha o nejrychlejší průchod (a tedy nepoužívání aktivní autentizace a autentizace čipu) vedly k řadě dohadů o nedostatečném zabezpečení těchto systémů.

Původní obavy o rychlost čtení dat z pasů (resp. skutečně nízká rychlost čtení pasů některých zemí) již vzaly za své. Moderní pas tak umí poskytnout všechna uložená data (kolem 48 kilobajtů) za 2,6 sekundy včetně všech bezpečnostních mechanismů (ano, i včetně provedení rozšířeného řízení přístupu).

Závěr

Zavedení elektronických pasů je možné označit za úspěch. Po technické stránce pasy fungují (v roce 1998, kdy se s ePasy začalo, to zdaleka nebylo tak jisté), bezpečnostní prvky ePasů se zlepšují, stejně tak i interoperabilita jednotlivých

zařízení a rychlost čtení. I v oblasti ePasů se ukázalo, že technické aspekty jsou ve většině případů snáze řešitelné, než aspekty organizační (a případně politické). Zvláště PKI infrastruktura pro EAC se rozjíždí velice pomalu, což snižuje aktuální využitelnost otisků prstů v pasech (kam se již přes dva roky ve většině zemí EU ukládají).

Literatura

[1] Bender, J., et al.: Security Analysis of the PACE Key-Agreement Protocol. *ISC 2009*, s. 33–48.

[2] Coron, J.-S., et al.: *Supplemental Access Control (PACE v2): Security Analysis of PACE Integrated Mapping.* Cryptology ePrint Archive, Report 2011/058, 2011. http://eprint.iacr.org/

[3] ICAO: *PKD Master Lists Contents.* Stažitelné z http://mrtd.icao.int.

[4] ICAO: *Public Key Direktory, PKD World Map.* Stažitelné z http://mrtd.icao.int.

[5] ICAO document 9303: *Machine Readable Travel Documents.* Stažitelné z http://mrtd.icao.int.

[6] ICAO: *Technical advisory group on machine readable travel documents (TAG/MRTD): Twentieth meeting*, Montréal, 7 to 9 September 2011, ePassports and biometrics. Stažitelné z http://mrtd.icao.int.

[7] ICAO technical report, *Supplemental Access Control forMachine Readable Travel Documents*, Version 1.01, November 11, 2010. Stažitelné z http://mrtd.icao.int.

[8] Kinneging, T.: *Supplementary Access Control*, NTWG TAG/MRTD 19, 19th Meeting of the Technical Advisory Group on Machine Readable Travel Documents. Stažitelné z http://mrtd.icao.int.

[9] Wikipedia: *heslo „Biometric passport“.* K dispozici na http://en.wikipedia.org/wiki/Biometric_passport.

[10] Williams, Ch.: *Facial recognition passport gates shut down. The telegraph.* 17 Feb 2011. K dispozici na http://www.telegraph.co.uk/technology/8330220/Facial-recognition-passport-gates-shut-down.html.

Správa revokovaných certifikátů v elektronickém platebním systému

Vít Bukač, Roman Žilka, Andriy Stetsko, Václav Matyáš

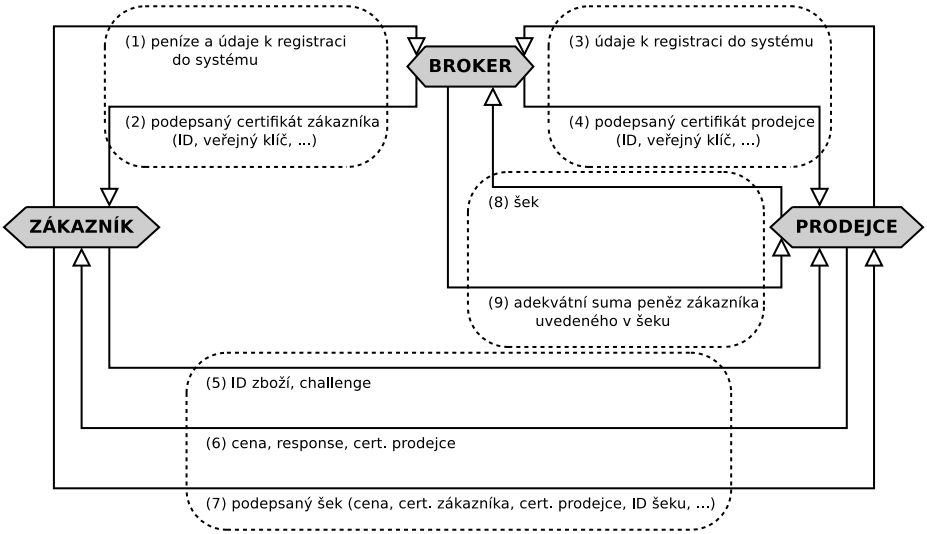
E-mail: bukac@ics.muni.cz, zilka@fi.muni.cz,
stetsko@fi.muni.cz, matyas@fi.muni.cz

Abstrakt

V moderním digitálním světě je běžné, že subjekty vlastní pár soukromý-veřejný klíč, pomocí nichž se autentizují svému okolí a provádí další úkony související s informační bezpečností. Veřejný klíč je často distribuován ve formě certifikátu, který tento klíč důvěryhodným způsobem pojí s identitou subjektu. Součástí životního cyklu certifikátu je také jeho revokace – odvolání informací v něm uvedených. V článku provedeme výklad vybraných současných metod správy seznamů revokovaných certifikátů (CRL – Certificate Revocation List) a představíme pro něj motivaci: otevřený systém pro elektronické obchodování, jehož bezpečnost je postavena mj. na certifikátech veřejných klíčů. V závěru diskutujeme vhodnost různých metod správy CRL pro uvedený platební systém.

1 Motivace pro téma

Zhruba před třemi lety se naše skupina z Laboratoře bezpečnosti a aplikované kryptografie na Fakultě informatiky Masarykovy univerzity pustila do vývoje otevřeného systému pro uskutečňování nízkonákladových nákupů pomocí čipových karet. Zadavatelem projektu je firma Y Soft, která dohlíží na skutečnou aplikovatelnost výsledků našeho snažení v praxi; loni se do spolupráce zapojil i Ústav práva a technologií Právnické fakulty MU. Cílem tohoto článku není představit projekt a přesvědčit čtenáře o jeho nedostižnosti, ale dotknout se dílčí problematiky projektu – správy *revokovaných* (tj. předčasně explicitně zneplatněných) elektronických certifikátů spojujících v našem případě mj. identitu subjektu (např. zákazníka) s jeho veřejným klíčem – a pokusit se vyvolat diskuzi na toto téma právě v kontextu celého připravovaného systému, jeho bezpečnosti a efektivity.



Obr. 1: Základní scénář užití platebního systému

Obrázek 1 zachycuje modelový chod cíleného platebního systému v největší obecnosti, bez kontrol nestandardních stavů (samozřejmě vč. samotné konzultace evidence revokovaných certifikátů, která není zatím implementována). V systému identifikujeme tři základní typy subjektů.

- *Zákazník*: fyzická osoba, která bezhotovostně – pomocí *čipové karty* či podobného platebního prostředku – nakupuje drobné položky (jednotky až nízké stovky Kč) jako např. občerstvení, parkovné či přístup k tiskárně na konferenci.

- *Prodejce*: firma, která nabízí zákazníkům zboží, za něž dostává výměnou digitální platidlo – *šeky* – které přiřknou finanční obnos rovný ceně zboží prodejci na vrub kupujícího zákazníka. V digitálním světě platebního systému prodejce zastupuje *prodejní terminál*, který je přítomný na prodejním místě a komunikuje se zákazníkovou čipovou kartou. Fakt, že prodejních míst může být pod správou jednoho prodejce více taktéž implicitně ignorujeme.

- *Broker*: firma, která manipuluje s penězi zákazníků, vydává jim čipové karty a je schopna proplatit digitální šeky za peníze v souladu s informacemi uvedenými v šeku: čí peníze v jakém množství mají být převedeny kterému prodejci. Brokera v systému personifikuje tzv. *zabezpečený terminál*.

Platí přitom, že meze jedné instance systému jsou dány dosahem jednoho brokera – v jednom systému může figurovat teoreticky neomezené množství prodejců a zákazníků a každý se smluvně dohodl s brokerem na zařazení do systému a podmínkách užívání. Tuto násobnost prodejců a zákazníků však pro účely tohoto článku, až na vyznačené výjimky, pomineme. Dále implicitně ignorujme fakt, že za prodejním, resp. zabezpečeným terminálem se může skrývat ne jedno hardwarové zařízení, ale celá infrastruktura, z níž jednotlivé části slouží ke komunikaci s vybranými subjekty za vybraných okolností.

Pravidla používání systému jsou řízena předně smlouvami, které broker uzavírá s každým prodejcem a zákazníkem a v užším pohledu pak *platebním schématem*. Pod tímto termínem chápeme pravidla (převážně elektronické) komunikace, kterou mezi sebou zástupci tří uvedených rolí uskutečňují a která zajišťují plnění účelu systému, a to za zachování co nejvyšší možné úrovně bezpečnosti. Platební schéma a jeho bezpečnostní mechanismy zde podrobně popisovat nebudeme; pro bližší informace viz [6] či podrobnější [5].

Jak bylo naznačeno, platební schéma využívá k zajištění určitých aspektů bezpečnosti asymetrické kryptografie a specifické formy certifikátů veřejných klíčů. Každý subjekt vlastní alespoň jeden platný (momentálně či v historii) certifikát a privátní klíč příslušný veřejnému klíči v certifikátu. Pro zákazníky a prodejce se tím myslí alespoň jeden certifikát vydaný brokerem. Náplň tohoto článku – revokace certifikátů – lze tedy označit za podmnožinu platebního schématu. Následující kapitola se bude věnovat detailům schématu, které jsou relevantní pro revokaci certifikátů. Kapitola třetí formou tématického exkurzu pojedná o některých metodách správy *seznamů revokovaných certifikátů* (*CRL*, *Certificate Revocation List*) a diskuzi vhodnosti vybraných metod konkrétně pro předložený projekt platebního systému pokryjí kapitoly 4 až 6.

2 Vybraná specifika platebního schématu

Přestože platební schéma operuje s certifikáty veřejných klíčů vlastněnými jak brokerem a prodejcem, tak zákazníkem, omezíme se v článku pouze na problematiku revokace zákaznických certifikátů. Situace kolem brokerova certifikátu je relativně jednoduchá: o případné revokaci se dozvědí jednorázově všichni prodejci smluveným způsobem přímo od brokera a přestanou přijímat šeky, neboť je broker již neproplatí. K žádné další komunikaci mezi žádnými dvěma subjekty v této věci nedochází. Revokace certifikátu prodejce je taktéž pouze jednorázově oznámena brokerem dotyčnému prodejci. Zákazníci o revokaci certifikátu prodejce explicitně uvědomováni nejsou – i kdyby náhodou došlo k tomu, že prodejce (uvědomený o revokaci svého certifikátu) se zákazníkem uskuteční obchod, zákazník získá zboží zadarmo, neboť šek za zboží si prodejce již nebude moci nechat proplatit. Obdobné zproštění zákazníka od starostí se aplikuje i v případě revokace certifikátu brokera.

Zákazník svůj certifikát iniciálně získává při registraci u brokera a přitom brokerovi buď svěřuje část svých financí, kterými pak bude broker bezprostředně disponovat, anebo se smluvně zavazuje zpětně závazky vůči brokerovi splácet. Certifikát je brokerem digitálně podepsaný a svazuje veřejný klíč zákazníka se sériovým číslem certifikátu, dobou platnosti a dalšími údaji. Sériové číslo je unikátní v celém systému od počátku existence systému a liší se i mezi certifikáty patřícími jednomu zákazníkovi. Jen broker a dotyčný zákazník umí spojit sériové číslo s ostatními (privátními a v certifikátu neuvedenými) informacemi o držiteli certifikátu.

Právě podle sériového čísla je certifikát také revokován a ihned po první revokaci se přestává používat s trvalou platností. Prodejci jsou motivováni nepřijímat šeky vystavené za použití revokovaného certifikátu platebním schématem – broker odmítne takové šeky proplatit. Řádná doba platnosti certifikátu se očekává v rozmezí 1–5 let; nelze však vyloučit vzácné scénáře s libovolně nižším rozmezím. Možné důvody k revokaci certifikátu před skončením tohoto rozmezí jsou následující:

- zákazník nedodržuje pravidla používání systému;
- jedna ze stran broker-zákazník rozváže vzájemnou smlouvu;
- smlouva mezi zákazníkem a brokerem se změní v takovém smyslu, že je třeba certifikát vyměnit (obzvl. změní-li se některý z údajů uvedených přímo v certifikátu);
- zákazník utratí (umožní-li to platební schéma pro daného zákazníka na základě jeho dohody s brokerem) víc peněz, než kolik brokerovi předem svěřil do opatrování;
- zásahem člověka či přirozeným opotřebením se objeví vada na fungování čipové karty zákazníka;
- zákazník explicitně požádá o revokaci svého certifikátu (např. z důvodu krádeže čipové karty).

Certifikáty, které přirozeně vyexpirují dosažením horního limitu platnosti, není třeba explicitně evidovat jako revokované. Je-li certifikát revokován během jeho přirozené doby platnosti, je třeba jej evidovat jako revokovaný nejméně do momentu jeho přirozené expirace. Revokovat certifikát může pouze broker na základě vlastního rozhodnutí nebo autentizované žádosti vlastníka certifikátu.

Prodejce může dle vlastní vůle neakceptovat libovolné další vybrané certifikáty interně; toto rozhodování však leží vně platebního schématu a i správa seznamu takto „blacklistovaných“ certifikátů je ponechána v režii prodejce. Prodejce se smluvně zavazuje brát ohled na seznam brokerem revokovaných certifikátů buď v jeho aktuálním znění, nebo ve znění ne starším než jistý interval.

Volba jedné z alternativ, stejně jako případné stanovení intervalu (v rozmezí 2 hodin až několika dnů) jsou předmětem individuální dohody.

Počítáme s jediným možným způsobem, který broker a prodejce mohou potenciálně použít pro komunikaci ve věci revokovaných certifikátů: zabezpečený (autentizace + integrita + soukromí zajištěné běžnou asymetrickou nebo symetrickou kryptografií) kanál v Internetu. Nicméně nespecifikujeme, jaké má kanál dosahovat datové propustnosti, latence a dostupnosti (vždy či jen občas).

3 Metody ověření platnosti certifikátu

Ve standardní situaci poslouží k rozhodnutí, zda dané certifikáty považovat za revokované či nikoli, seznam revokovaných certifikátů vydaný příslušnou *certifikační autoritou* (*CA*, *Certificate Authority*). Pokud však není možné revokační status certifikátu ověřit (např. je nedostupný server autority nebo vypršela platnost dostupných CRL), jsou k dispozici dvě strategie, jak revokační status certifikátu stanovit. *Liberální* strategie říká, že certifikát je považován za platný, pokud není (prostřednictvím CRL) doložen opak. Je tedy prosazována dostupnost před bezpečností, neboť certifikát, o kterém nevíme, zda byl revokován, je přijat. U *konzervativní* strategie považujeme certifikát naopak za neplatný, pokud není doložen opak. Je tudíž prosazována bezpečnost před dostupností.

3.1 Kompletní CRL

Seznam revokovaných certifikátů je digitálně podepsaná, pevně definovaná struktura, která obsahuje minimálně [2, 3]:

- číslo verze formátu CRL;
- identifikátor podpisového algoritmu CRL;
- název certifikační autority, která uveřejňuje CRL;
- datum a čas uveřejnění;
- datum a čas skončení platnosti;
- sériová čísla revokovaných certifikátů;
- datum a čas revokace každého certifikátu;
- hodnotu digitálního podpisu.

Seznam revokovaných certifikátů je periodicky uveřejňován certifikační autoritou. Při každém ověření platnosti certifikátu je nutné ověřit, že sériové číslo

certifikátu není uvedeno v seznamu revokovaných certifikátů příslušné certifikační autority. Při prvním ověření platnosti ověřující entita CRL stáhne a uloží ve své lokální paměti. Následující ověření již probíhají proti této paměti. Tím je minimalizována doba ověření a zátěž pro certifikační autoritu. Po uplynutí času platnosti je nutné stáhnout nový CRL. Důsledky vyplývající z neochoty nebo neschopnosti stáhnout nový CRL je možné řešit smluvně. Například broker nese vinu za transakce provedené s kompromitovaným certifikátem v době mezi kompromitací a zveřejněním čísla certifikátu na CRL, zatímco prodejce nese vinu za transakce provedené až po zveřejnění.

Revokací certifikátu je v podstatě myšleno umístění jeho sériového čísla do nového periodicky uveřejňovaného CRL. V době mezi požadavkem na revokaci a zveřejněním nového CRL je certifikát při ověření označen stále za platný. Tato doba může mít až délku periody mezi uveřejněními dvou CRL – např. 1 den. Seznam revokovaných certifikátů je digitálně podepsán vydávající certifikační autoritou, aby se zabránilo jeho podvrhnutí. Certifikáty jsou ze seznamu revokovaných certifikátů vyřazeny nejdříve po uplynutí data jejich konce platnosti.

Existují dva režimy předání CRL mezi certifikační autoritou a ověřující entitou. V *pull režimu* zasílá ověřující entita certifikační autoritě žádost o předání seznamu. CRL je zaslán v odpovědi na žádost. CA musí být v každém okamžiku dostupná a připravená CRL odeslat. Pull režim je běžný ve většině současných systémů. V *push režimu* rozesílá CA nový seznam všem ověřujícím entitám okamžitě po jeho uveřejnění. Mezi časy uveřejnění seznamů může být CA offline, musí si však udržovat seznam všech ověřujících entit. Režimy je možné i kombinovat, například prvotní zveřejnění CRL provést v push režimu a pull režim využít pro nové nebo dočasně nedostupné ověřující entity.

3.2 Delta CRL

Delta CRL obsahují pouze sériová čísla těch certifikátů, které byly zneplatněny po určitém okamžiku v čase (typicky po zveřejnění jiného CRL) [1].

- *Inkrementální delta CRL*: delta CRL obsahuje sériová čísla certifikátů revokovaných od uveřejnění předchozího CRL. Tím může být kompletní CRL nebo jiný delta CRL. K ověření certifikátu je nutné získat prvotní kompletní seznam revokovaných certifikátů a všechny následující delta CRL.
- *Rozdílový delta CRL*: delta CRL obsahuje sériová čísla certifikátů, které byly revokovány od uveřejnění posledního kompletního seznamu. K ověření certifikátu je nutné získat prvotní kompletní seznam revokovaných certifikátů a poslední delta CRL.

Použití delta CRL umožňuje prodloužit periodu uveřejňování kompletních CRL i při zachování nebo dokonce zkrácení maximálního stáří revokační infor-

mace. Tím může být snížena zátěž sítě a urychlen proces ověření. Delta CRL mohou být ukládány do paměti obdobně jako kompletní CRL. U systémů s nepříliš častými ověřeními platnosti můžeme zákazem ukládání do paměti u rozdílových delta CRL zajistit dostupnost revokační informace v reálném čase, a to i při nízkých nárocích na síť.

3.3 Online Certificate Status Protocol

Online Certificate Status Protocol (*OCSP*) je protokol typu výzva-odpověď navržený pro rychlé ověření platnosti certifikátu v reálném čase [4]. Certifikační autorita musí být neustále online; ověřující entita musí být schopna se připojit na vyžádání. Při každém požadavku na ověření certifikátu zasílá ověřující entita žádost certifikační autoritě. Tato žádost obsahuje hlavně časové razítko a seznam sériových čísel ověřovaných certifikátů. Žádost může být digitálně podepsána. V odpovědi přidá certifikační autorita ke každému sériovému číslu stav certifikátu (nerevokovaný/revokovaný/neznámý), případně i datum a čas revokace. Odpověď je digitálně podepsána, aby se zabránilo jejímu podvrhnutí. Samotné rozhodnutí o přijetí či nepřijetí certifikátu leží stále na ověřující entitě. Protokolem OCSP nejsou ověřovány žádné další faktory nutné pro rozhodnutí o platnosti certifikátu (např. zda již neuplynula doba jeho přirozené platnosti).

Při použití OCSP mohou být sníženy nároky na rychlost přenosového média. Nevýhodou jsou vyšší nároky na výpočetní výkon na straně certifikační autority. Každá odpověď na žádost o ověření vyžaduje provedení jedné až dvou kryptografických operací (ověření podpisu žádosti a podepsání odpovědi).

4 CRL v kontextu platebního systému

4.1 Scénáře ověření certifikátu

Pro naši analýzu jsme definovali sedm různých scénářů ověření certifikátu. Scénáře 1 až 3 využívají pouze základní seznamy revokovaných certifikátů. Scénáře 4 až 6 využívají základní seznamy doplněné o delta seznamy. Posledním scénářem je použití protokolu OCSP.

Scénáře byly zvoleny tak, aby maximální stáří revokační informace bylo v rozmezí 15 minut až 24 hodin. Domníváme se, že v námi popisovaném systému není nutné vyžadovat větší čerstvost než 15 minut, neboť se mohou vyskytovat krátké výpadky přenosových médií, kterým nelze zabránit. Také nesmíme zapomenout na případný rozdíl časů mezi okamžikem, kdy dojde ke kompromitaci soukromého klíče zákazníka (např. ukradení čipové karty), zjištění kompromitace uživatelem (např. odhalení chybějící peněženky) a skutečnému vznesení požadavku na revokaci. Rozdělení zodpovědnosti za případné ztráty v každém období může

být specifikováno ve smlouvách mezi brokerem, prodejcem a zákazníkem. Také nepředpokládáme situaci, kdy by bylo třeba pracovat s revokační informací starší než 24 hodin. Jeden den je obvyklá nejzazší mez v obdobně fungujících systémech a její vynucování představuje jen malou zátěž.

1. CRL15: každých 15 minut je uveřejněn kompletní CRL.
2. CRL120: každé 2 hodiny je uveřejněn kompletní CRL.
3. CRL1440: každých 24 hodin je uveřejněn kompletní CRL.
4. CRL240+15: každé 4 hodiny je uveřejněn kompletní CRL; každých 15 minut je uveřejněn inkrementální delta CRL.
5. CRL1440+15: každých 24 hodin je uveřejněn kompletní CRL; každých 15 minut je uveřejněn inkrementální delta CRL.
6. CRL1440+120: každých 24 hodin je uveřejněn kompletní CRL; každé 2 hodiny je uveřejněn inkrementální delta CRL.
7. OCSP: ověření pomocí Online Certificate Status Protocol.

U každého scénáře byly posuzovány požadavky na datovou šířku pásma, počet kryptografických operací a celkové množství přenesených dat.

4.2 Předpoklady systému

Srovnání jednotlivých scénářů bylo provedeno za následujících podmínek.

- Kompletní seznam revokovaných certifikátů má velikost 300 KB. V souboru této velikosti se uloží záznamy o zhruba 8 000 revokovaných certifikátech – viz [3]. Například VCA2 seznam revokovaných certifikátů české kvalifikované certifikační autority PostSignum z 8. června 2011 má ve formátu DER velikost 56 KB a nese záznamy o 1 565 certifikátech.
- Povinná pole CRL (např. hodnota digitálního podpisu, časová známka) zabírají minimálně 500 B.
- Delta CRL, který je uveřejněn každých 15 minut, má velikost 1 KB.
- Delta CRL, který je uveřejněn každých 120 minut, má velikost 6 KB.
- Obvyklá odpověď na požadavek ověření platnosti certifikátu protokolem OCSP má velikost 250 B [4].

Tab. 1: Technologie přenosu dat dle tříd

	Datový tok	Technologie	Čas	Latence
Minimální	0–1 KB/s			
Velmi malý	1–10 KB/s	Dialup	60 s	∼100 ms
		GPRS	40 s	100 ms–1 s
Malý	10–100 KB/s	EDGE	15 s	100 ms–1 s
Střední	100 KB/s–1 MB/s	802.11b	1 s	10–100 ms
		CDMA	1 s	∼100 ms
		HSDPA	1 s	∼100 ms
Velký	1–10 MB/s	802.11a/g/n	1 s	10–100 ms
		ADSL	1 s	∼10 ms
		Ethernet	1 s	∼10 ms
Velmi velký	10–100 MB/s	Fast Ethernet	1 s	∼10 ms
Extrémní	> 100 MB/s	Gbit Ethernet	1 s	∼1 ms

- Na každého prodejce připadá v průměru 500 zákazníků. Každý zákazník provádí v průměru 10 transakcí za den, rovnoměrně rozložených mezi všechny prodejce. Jeden prodejce tedy provede v průměru 5 000 transakcí denně.
- Šíření CRL probíhá v pull režimu. Jednotliví prodejci si stahují revokační informace podle aktuální potřeby.

5 Posouzení metod ověření platnosti certifikátu

Spočítané hodnoty datových toků budeme pro srozumitelnost dělit do 7 tříd (viz tabulka 1). U každé třídy jsou uvedeny i síťové technologie, jejichž maximální teoretická rychlost do ní spadá. Sloupec „Čas“ uvádí, jak dlouho v průměru při použití dané technologie trvá stažení souboru s velikostí 300 KB. Sloupec „Latence“ uvádí řádové hodnoty zpoždění dané technologie.

Vysoká latence, typická hlavně pro bezdrátové mobilní technologie (GPRS, EDGE, HSDPA, CDMA), může být v některých situacích velmi limitující pro použití protokolu OCSP (např. platbu za vstup do dopravního prostředku je nutné provést obvykle do vteřiny).

Tabulka 2 shrnuje základní parametry každého scénáře. První sloupec udává délku maximální doby, po kterou může být uznán za platný certifikát, u kterého již byl vznesen požadavek na revokaci. Druhý sloupec udává teoretický počet špiček z hlediska nároků na přenosovou kapacitu certifikační autority. V případě

Tab. 2: Základní vlastnosti scénářů

	Max. čas chybné přijetí	Max. počet špiček za den
CRL15	15 min	≤ 100
CRL120	2 h	≤ 12
CRL1440	24 h	≤ 1
CRL240+15	15 min	≤ 6
CRL1440+15	15 min	≤ 1
CRL1440+120	2 h	≤ 1
OCSP	0 min	Variabilní

Tab. 3: Množství dat stažených každým prodejcem za časový interval

	1 h	4 h	1 den
CRL15	1,2 MB	4,8 MB	28,8 MB
CRL120	—	0,6 MB	3,6 MB
CRL1440	—	—	0,3 MB
CRL240+15	—	0,32 MB	2 MB
CRL1440+15	—	—	0,4 MB
CRL1440+120	—	—	0,4 MB
OCSP	0,05 MB	0,2 MB	1,2 MB

použití kompletních i delta CRL je za špičku považován okamžik, kdy více jak polovina všech prodejců najednou zažádá o zaslání kompletního CRL. Denně tedy může být maximálně tolik špiček, kolik je vydáno kompletních CRL. Reálný počet špiček bude obvykle výrazně nižší (např. v noci k nim pravděpodobně docházet nebude). Počet špiček u protokolu OCSP není možné přesně stanovit – řídí se převážně charakterem poskytovaných služeb (např. prodej novin se špičkou brzy ráno vs prodej jízdenek do MHD se špičkou v odpoledních hodinách).

V případě scénářů s delší periodou uveřejňování kompletních CRL (CRL1440, CRL240+15 atd.) je možné naplánovat časy tak, aby spadaly mimo období s nejčastějšími požadavky. Tím dosáhneme rovnoměrnějšího rozložení zátěže.

Tabulka 3 popisuje množství dat stažených jedním prodejcem za jednotku času. Do hodnot nejsou započítána řídící data (hlavičky IP, řídící segmenty TCP apod.) ani odeslaná data. V případě, že se provádí platby za přenesená data, je nejméně výhodné použití kompletních seznamů s krátkou periodou uveřejnění. Použití delta CRL a OCSP žádný problém nepředstavuje.

Tabulky 4 a 5 uvádějí počty kryptografických operací prováděných brokerem resp. prodejcem každý den. V případě kompletních CRL a delta CRL se jedná o maximální počty, reálná čísla mohou být nižší v závislosti na frekvenci

Tab. 4: Počet kryptografických operací prováděných brokerem každý den

	< 100 prodejců	100–1 000 prodejců	> 1 000 prodejců
CRL15	< 100	< 100	< 100
CRL120	12	12	12
CRL1440	1	1	1
CRL240+15	< 100	< 100	< 100
CRL1440+15	< 100	< 100	< 100
CRL1440+120	12	12	12
OCSP	500 000	5 000 000	> 5 000 000

Tab. 5: Počet kryptografických operací prováděných prodejcem každý den

	Kryptografických operací za den
CRL15	< 100
CRL120	< 12
CRL1440	1
CRL240+15	< 100
CRL1440+15	< 100
CRL1440+120	< 12
OCSP	5 000

vydávání CRL, četnosti požadavků na revokaci a rozložení těchto požadavků v čase. U OCSP je počet kryptografických operací roven počtu transakcí. Pokud jsou žádosti OCSP digitálně podepsány a broker tyto podpisy ověřuje, je počet prováděných kryptografických operací roven dvojnásobku počtu transakcí.

Počítání mnoha kryptografických operací je hlavním omezením protokolu OCSP. Rozsáhlé a náročné systémy, kde by zátěž při používání pouze protokolu OCSP byla příliš velká, mohou využívat hybridní přístup. Na většinu transakcí jsou používány (delta) CRL, pouze podmnožina je ověřována přes OCSP. Které transakce využívají OCSP, je na dohodě mezi CA a ověřujícími entitami – mohou to být například transakce od určité hodnoty. Hybridní systémy mohou také zvyšovat dostupnost služby. V takovém případě je primárním způsobem ověření OCSP, ale prodejce i přesto na pozadí stahuje CRL. Pokud dojde ke ztrátě spojení s brokerem, může prodejce přejít na ověření pomocí CRL, které má již uložené v lokální paměti.

Tabulka 6 udává průměrné nároky na rychlost síťového připojení brokera. Hodnoty v ní uvedené jsou doporučeným minimem pro přijatelnou práci systému. Ve špičkách a při mnoha souběžných požadavcích je však minimum obvykle překročeno.

Tab. 6: Průměrné nároky na šířku přenosového pásma brokera

	< 100 prodejců	100–1 000 prodejců	> 1 000 prodejců
CRL15	Malý	Střední	Střední+
CRL120	Velmi malý	Malý	Malý+
CRL1440	Minimální	Velmi malý	Velmi malý+
CRL240+15	Velmi malý	Malý	Malý+
CRL1440+15	Minimální	Velmi malý	Velmi malý+
CRL1440+120	Minimální	Velmi malý	Velmi malý+
OCSP	Velmi malý	Malý	Malý+

Tab. 7: Maximální nároky na šířku přenosového pásma brokera

	< 100 prodejců	100–1 000 prodejců	> 1 000 prodejců
CRL	Velký	Velmi velký	Velmi velký+
OCSP	Malý	Střední	Střední+

Tabulka 7 udává teoretické nároky na rychlost síťového připojení brokera ve špičce. Rychlost stahování prodejci je v tomto případě považována za neomezenou. U CRL se předpokládá situace, že všichni prodejci zároveň si ihned po uveřejnění nového kompletního CRL zažádají o jeho zaslání. Každý prodejce musí dostat CRL nejpozději 5 vteřin po odeslání požadavku. OCSP vychází z předpokladu, že každý prodejce provádí nejvýše jednu transakci za vteřinu.

Při porovnání tabulek 6 a 7 je vidět, že pro námi nastavené podmínky dochází u CRL ke značnému nárůstu požadované šířky pásma při uveřejnění nového seznamu. V systému, ve kterém dochází velmi často k transakcím nebo jsou transakce nárazové, se hodnoty blíží maximu. V takové situaci je totiž velice pravděpodobné, že se objeví požadavek na transakci u každého prodejce velmi brzo po uveřejnění nového CRL. OCSP má průběh vyrovnanější, závislý zejména na celkovém počtu transakcí. Když by prodejci prováděli více transakcí souběžně, mohlo by dojít k nárůstu, ale zadání společnosti Y Soft toto nepředpokládá. Z hlediska datového toku je prodejce provádějící 2 transakce najednou ekvivalentní dvěma prodejcům ověřujícím sekvenčně.

6 Závěr

Naše výsledky ukazují, že ani v jednom případě nedosáhly nároky na přenosovou rychlost brokera extrémních hodnot. Moderní server s Gigabit Ethernet připojením by měl být schopen naplnit požadavky i velmi rozsáhlého systému. Škálovatelnost systému při použití pouze OCSP naráží na vysoké počty prováděných kryptografických operací.

Poděkování

Setkání řešitelského týmu a spolupracovníků se odehrává tradičně formou brainstormingových sezení. Za firmu Y Soft se jich účastní Ondřej Krajíček, tamější šéf výzkumu a rozvoje, který nejen že dohlíží na reálnou užitnost průběžných výsledků, ale rovněž aktivně zasahuje do vývoje systému jako odborník na různá dílčí technická a právní témata. Děkujeme Ondrovi za jeho nápady, které se obrazily i v textu tohoto příspěvku.

Literatura

[1] Adams, C., a Lloyd, S.: *Understanding PKI: Concepts, Standards, and Deployment Considerations*. 2nd Ed., Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0-672-32391-5.

[2] Cooper, D. A.: A Model of Certificate Revocation. In *Proceedings of the 15th Annual Computer Security Applications Conference* (1999), IEEE Computer Society, str. 256–. ISBN 0-7695-0346-2. URL (v květnu 2011): http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.1641&rep=rep1&type=pdf.

[3] Housley, R., Polk, W., Ford, W., a Solo, D.: *Internet X.509 Public Key Infrastructure, Certificate and Certificate Revocation List (CRL) Profile.* Request for Comments 3280, The Internet Society, duben 2002. URL (v květnu 2011): http://tools.ietf.org/html/rfc3280.

[4] Myers, M., Ankney, R., Malpani, A., Galperin, S., a Adams, C.: *X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP.* Request for Comments 2560, The Internet Society, červen 1999. URL (v květnu 2011): http://tools.ietf.org/html/rfc2560.

[5] Žilka, R.: *Towards a Secure Payment System for Large Infrastructures.* Rigorózní práce, Masarykova univerzita, Fakulta informatiky, červen 2010. URL (v květnu 2011): http://is.muni.cz/th/73058/fi_r/rigorozni_prace.pdf.

[6] Žilka, R., Tuček, P., Matyáš, V., a Stetsko, A.: Otevřené mikroplatební schéma pro rozsáhlé infrastruktury. In *Sborník příspěvků z 36. konference EurOpen.CZ* (květen 2010), V. Rudolf, Ed., EurOpen.CZ, str. 63–80. ISBN 978-80-86583-19-8. URL (v květnu 2011): http://www.europen.cz/Anot/36/eo-1-10.pdf.

Modern ways to design fully distributed, decentralized and stealthy worms

Norbert Szetei

E-mail: norbert.szetei@nethemba.com

1 The analysis of existent worms

Our first goal was to analyze the most intrusive botnets and worms in history. Because the several dangerous worms lack of clever architecture (Morris's worm, Love Letter) or they need for spreading the user interaction, we choose the representative worm from a distinct types and we describe the properties and techniques which they use. The first one, Waledac is a "mature" version of famous Storm Botnet that try to be more secure with the little different architecture. Warhol is a special type of worm for the extremely rapid propagation through the network. Finally, Conficker is a self-updating worm, his lately version uses P2P communication and propose Domain Generation Algorithm for different purposes.

The first generation of botnets uses IRC or instant messengers and lack of cryptography. Because everyone who get access to C&C can disable the whole network, using the P2P architecture which brings many advantage, but also presents the new problems.

1.1 Waledac

A little similar model like Storm that came to attention at late of 2008. It was coded in C++ and compiled with Microsoft Visual C++ compiler [1]. Like Storm, it is passive worm that replicate using emails that point to URL that is a part of Fast-Flux network. It uses P2P network for communication over HTTP with XML messages exchanging.

Waledac uses Spammer and Repeater nodes with different purposes. The subcontrollers are known as TSL servers (named as windows registry used to store list of peers for this layer) and the forward traffic between repeaters and Upper-Tier Servers (UTS) that is communicating with C&C server, see figure 1.

The decision if the infected computer will be Spammer or Repeater is made according IP address of host, Spammer is used for private range (192.168.0.0/16,

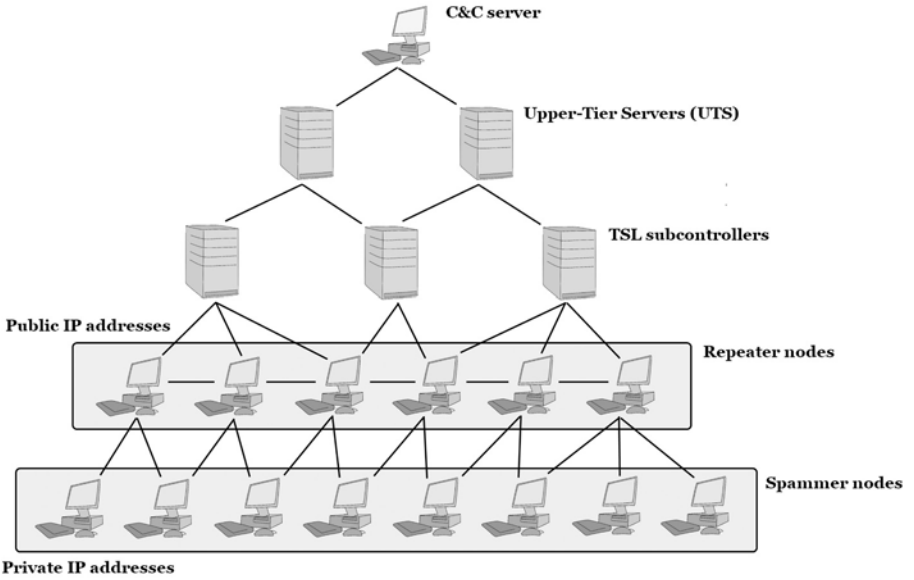


Figure 1: Waledac Architecture Layout

172.16.0.0/12, 10.0.0.0/8) and his purpose is as the name state to receive commands for spamming or performing DoS attacks and interaction with repeater nodes in a hierarchy immediately above. Researchers determined, that spammers accept arbitrarily valid IP address for repeaters list. They also do not now any information about location of other spammers and communicate only with the repeaters.

The repeaters uses public IP addresses and are the part of Double Flux network so they DNS name are rapidly changing. After they connection, Waledac ask the neighbour repeaters for the current TSL server list. If it is turned on the firewall on the system to be used as the Repeater, Waledac modifies settings for ensuring a smooth transition.

TSL server are formed mainly by nginx web server, according to [2] them have been identified five, two in Germany, one in the U.S., Russia and the Netherlands, and they provides common web hosting functionality.

Each node in the network contains a list of repeaters that determines, which nodes can be allowed for communication. Similarly repeaters contains a list of TSL servers.

The Waledac binary contains a list of IP addresses for bootstrap and make an initial connection with botnet. Firstly after connection of new node, it tries to contact the repeater. In the case, that node is unable to find any active node in list, it has hard coded URL that try to reach, protected by Fast Flux.

Waledac binary also contains hard coded password. After locating the repeater node, encrypted HTTP request is sent to him with public key certificate generated at startup. This certificate is propagated through layers above to the C&C server and it is returned the actual communication key encrypted with new node key. To any additional communication with neighbours nodes are used this new key. It appears as programming error that the key is always the same. For various purposes and between nodes of different functionality, Waledac uses Base64 encoding with AES, RSA, SHA1 and BZIP2 algorithms.

Repeaters maintains list of active nodes in network. Each node has 500 or 1 000 entries each entry contain the timestamp per IP address, so repeater node can remove the old entry and to have the list up to date. Waledac did not have DHT architecture used by Storm and it uses random peer selection. Also this layer (repeaters tier) is the only really P2P network. Spammers have limited external connectivity and TSL and every layer above serves primarily as proxy for master command and control.

The researchers tried the integration to appropriate high layer, but was enable to enumerate the number of spammers, what was relatively easy in Storm's Overnet protocol. From repeater layout they will only determined the small subset of overall botnet and to date, no one was able to accurately identify the size of whole network.

1.1.1 Defense mechanism

TLS update is the special command used by repeater to update its TLS servers list. Not like spammers, the repeaters did not accept only verified IP in two steps:

- Every entry contains timestamp and old entities are refused
- Entry is digitally signed with RSA, each Waledec binary contains a copy of the public key and unauthorized entities are refused

So if the attacker try to carry out "reply attack" and alter entries with the old ones, they will be refused.

1.1.2 Protection against Waledac

There are some different methods how to mitigate Waledac botnet:

1. Unlike Storm, Waledec does not attempt to be stealthy and uses fixed location on disc (My Documents or Desktop directory), so it is easily detectable.

2. If we connect our repeater to the network, we can be informed about Fast Flux network domains. Blocking these domains can little reduce the infection.

3. Repeater node update function depends on string "X-Request-Kind-Code: nodes", that is sent as plaintext, so IPS/IDS can exactly match it.

1.1.3 Waledec's Takeover

In [2] was presented the technique how to disable the whole botnet. The first step is injecting one or more repeaters to botnet, which we named "sinkhole". Receiving the instructions from command and control, they do not carry out any operation and any request for node list will results in providing the faked arbitrary IP list with random hashes. These hashes should not be a part of botnet, because Waledec uses IP addresses only in first time and another update uses hash value and could to reinforce any valid malicious node. After the majority of spammers and repeaters points to the fake nodes, attack continue to the next stage to taking control over Fast Flux network.

Fast Flux domain names are hard coded to binary package, so we can poison the domain database with own entries which point to fake nodes and remove .com TLD zone. Last step is to disable all current TSL servers by ISP. The timing is critical and all TSL should be disabled at the same time. After fake nodes expiration, because a malfunction of TSL, the botnet will be unable to recover and botnet master lost the control over repeaters and spammers.

1.2 Conficker

Conficker (also known as Downadup) worm was firstly detected in November 2008 and since SQL Slammer worm in 2003, it is deemed that it caused the largest computer worm infection with few millions IP addresses in 206 countries. It had several mutations and variants where the code drastically changed, we use naming from The Conficker Working Group (see later) as A, B, B++, C, and E as in [3].

1.2.1 A, B, B++ variants

The early version of Conficker (A sample) exploits the MS08-067 vulnerability over port 445/TCP in Windows Server service which cause possibility for running arbitrary code without authentication. The computer researchers states that there were five Conficker variants A-E with different injection vectors and functionality.

After computer infection, Conficker sample A's agent firstly runs a HTTP server on the infected computer before and from newly infected victim computer, he retrieve an worm replica DLL. It also attempts to download the GeoIP database from maxmind.com and that use for scanning the new victims for filtering IP addresses of Ukraine. Furthermore it checks the keyboard layout on infected computer and when it is set to Ukraine, the worm commits a suicide.

The most interesting thing on Conficker is DGA (domain generation algorithm). Conficker A in the last phase enters to infinite loop and with seed derived from current time (synchronized to the current UTC), he generates the list of 250 domain names. When the valid IP address is found, it is contacted and agent try to downloads the windows binary from this domain (rendezvous point) and immediately after then to validates digital signature with public key attached to every sample. If the hash integrity succeeds, the binary is executed via shellexec() call. Conficker with this algorithm tries to validate every IP from generated list and then to sleep next three hours, so it iterates the generation process 8 times in one day.

Conficker's agent with sample B detected in January 2009 did not check the presence of Ukraine keyboard. It generates domains for rendezvous point every 2 hours and with an little difference, so domains do not overlap. It also uses more sophisticated techniques for propagation as dictionary attacks with 240 common passwords to NetBIOS share or removable USB media with autorun files. Lately mutation of B variant also provides to download and with valid signature to run the arbitrary EXE file, not only DLL. It can also especially patch vulnerability to prevent to the others to exploit infected machine, but not for the instances of Conficker agent.

With the growing infection, the number of companies with antivirus industries grouped together and formed the "Conficker Working Group"[1] and they cooperate with Internet Domain Registrars to block the domains, that attempts the worm use for communication or updates purposes. But because in that time the worm had the P2P architecture, this thing was still possible. This domain was also used for monitoring of wormnet growing and activity.

1.2.2 C variant

Lately in February 2009 the C variant allowed to receive URL from remote hosts and to use for download. It leaves just 15 % of code from A or B samples as an answer for mitigation tried by antivirus companies.

It copies itself with randomly named DLL file that can be located in "System32" directory, "Program Files" or user's temporary files folder.

[1] http://www.confickerworkinggroup.org

Felix Leder and Tillman Werner revealed [4] that C variant contains hard coded date, when the "worm" activates and start to look after binary updates. That was after 1.4.2009. There were a speculations if it was only an April Fool Day joke, and receives also an public attention, so Conficker Working Group prepared for this event, but nothing serious happened that day.

C sample had a sophisticated adoption of advanced hash algorithms, binary encryption and detection of virtual machines. With P2P module, there were now three ways for self-update:

- rendezvous point with extended DGA
- re-exploitation of MS-067 vulnerability
- P2P network

Conficker's authors used for worm protection RSA, RC4 and MD-6 algorithms. The interesting thing is, that MD-6 had been announced only a few weeks before the algorithm was found in Conficker, so the author was aware of the latest news in cryptography.

Also Conficker uses secure distribution and update mechanism and according study of its code the researchers suppose that it is a work of security specialist with a deep knowledge of Windows internals.

The core parts of C are two threads, one for domain generation algorithm and one for peer to peer network.

1.2.3 Domain Generation Algorithm

The most recent known version of Conficker C uses DGA that generates about 50,000 potential domains after activation date (1.4.2009). From this, only 500 were queried and only once per 24 hours with difference of previous models.

Every node tried to resolve IP address from domain set. The IP address is rejected if the one of the following conditions is fulfil with a returned IP address:

- DNS request returned more than one IP address
- It was localhost or something in internal network or IP address from internal blacklist
- Another query returned this IP address, so there were multiple domains associated with it

Than it made a "GET" requests to appropriate web server IP addresses and tried to verify digital signed (RSA or MD6) payload. If this fails, the worm sleeps for another 24 hours and repeat the whole process. For internal time synchronization, the C uses random one of the list of the web servers and parse the time

from HTTP response header. Synchronization web server list includes the important sites like adobe.com, ebay.co.uk, facebook.com, megaupload.com, rapidshare.com, seznam.cz, wikimedia.org, yahoo.com, youtube.com, so disabling the servers is very hardly possible.

1.2.4 Peer to peer mechanism

P2P mechanism two TCP and UDP ports and obfluscation with anti debugging logic. Every node has download and upload functionality. Unlike Storm wormnet, *Conficker C has uniquely assigned TCP/UDP ports for each IP address, so there is no need for supernodes and distribution of peer list.* Every instance of worm agent can do random scanning and join to the appropriate peers without any information about bootstrap nodes.

1.2.5 Security protection disablement

For hiding its presence, Conficker disables security products that could possible detect it. Also C variant disables firewall for P2P peers connection and Security Defender and Update Services in Microsoft Windows operation systems. Finally, it deletes all of the Windows Restore points and prevents domain lookups for the most security products. If they monitor the presence of some rootkit detector or packet sniffer, immediately kill this process.

1.2.6 Conficker's mitigation

Felix Leder and Tillmann Werner on 27 March 2009 discovered, that Conficker has detectable signature and can be remotely scanner with essential security tools like nmap or nessus scanner. Microsoft also provides Windows Malicious Software Removal Tool for worm disinfection and system security patch.

1.3 Warhol Worm

Warhole is a very old and an extremely dangerous type of worm that can spread as fast as possible through physical resources usually from single node with the following algorithm:

1. Scan for machines on the Internet for specific ports
2. Determine the running service
3. Send a probe to infect the target
4. If it was successful, transfer a copy of itself to new target
5. Repeat the process for each infected nodes

The scanning method is very important and should be pseudo random, the worst case is to scan linear address range, which can be trivially detected by Intruder Detection Systems (IDS) or Intruder Prevention Systems (IPS).

The recommended way is to use 32 bit long block cipher and for generating the i-th entry, we just encrypt i. Equally if we need the index, we just decrypt i. The encryption key can be hard coded in the worm and his disclosure has no impact for a worm mitigation. The redundant generated IP addresses can by eliminated using a coordinated worm (with communication) model.

Because the spreading is exponentially fast, it is the most effective to have an initial hit list. The attacker can scan the whole network in weeks before the worm launching and obtain a list with vulnerable services. With only 5 000 machines the total time for infection the most of vulnerable machines can be cut to the few minutes with no coordinated worm, coordinated worm can be even faster.

2 Worm proposal

Because botnets and worms carry out nefarious tasks such as a spam campaign, distributed denial of service attacks and information theft, we have dedicated a significant effort to research in order to better understand them.

We described dangerous worms from the past that were mitigated years ago. Current worms generally use P2P network instead of early central C&C centers and have additional layers of cryptography for anonymity, securing the communication and polymorphism aspects.

If we want to propose a new worm, firstly we need to consider the purpose of them. We need to define the working environment and more importantly, the longevity. If we want to write the worm for a short time to accomplish one specific task, we do not need any coordination. In this case, Warhol model is probably still the best way to accomplish.

2.1 Short-time worm

There is an easy way to download some exploits and use them for worm replication.

To ensure that the IP addresses are repeated as least as possible we have used nmap[2] source code which included Congruential Linear Generator (LCG) for a random IP address generation (-iR parameter). We consider that we have 100 infected computers and are able to scan the 1 million IP addresses from this computer, we have created the diagram of the increasing number of unique IP addresses.

[2] http://www.nmap.org

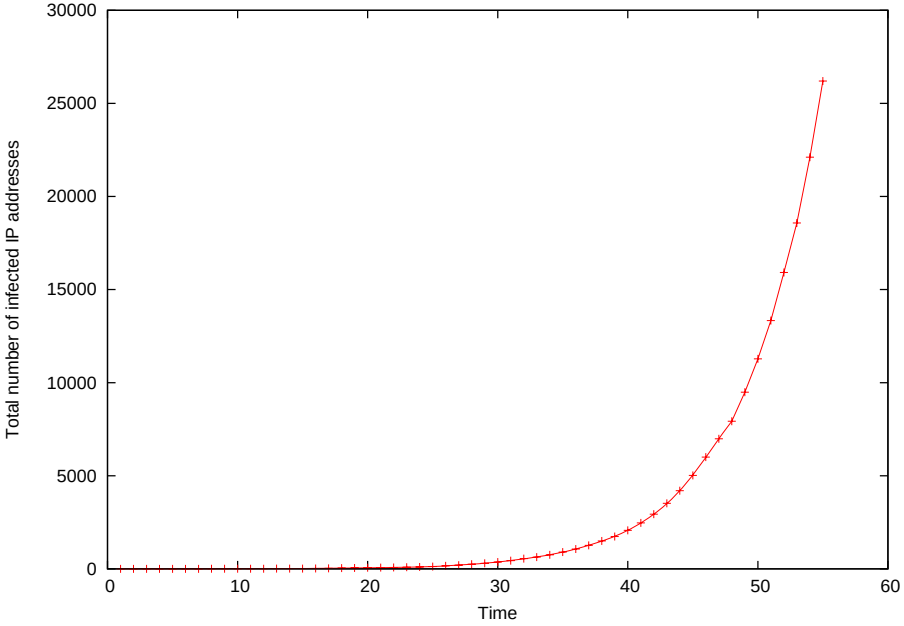


Figure 2: Apache 2.2.3 worm spreading visualisation

Because we have detected only small number repetitions for a huge number of gen- erated IP addresses, we consider this generation algorithm to be very sufficient. For testing purposes we have written the regular expression scanner that grabs banners according to the given regular expressions on specific ports and save all matched IP addresses. We have tried to simulate the worm spreading for vulnerable ProFTP daemon[3], but stopped it after our ISP complaints.

Due to the fact that there is a lot of HTTP traffic caused by Internet crawlers like GoogleBot or YahooBot, we have continued in scanning of web services (HTTP protocol) which was totally acceptable by our ISP.

We have chosen to visualize our worm propagation for the Apache Web Server 2.2.3 which is not very widespread and according to our opinion sufficiently representative. During our simulation we have found the first computer responding to our request with the banner "Apache/2.2.3" that was labeled as infected. We ran multiple simulations in the sequential order in order to look "parallel" (because we have restrictions on transmission lines, we have conducted a simulation in linear time and scale as the parallel model) scan and labeled "infected" nodes, resulting in an exponential function characteristics, as we can see in the diagram/picture 2.

[3] http://lists.grok.org.uk/pipermail/full-disclosure/2010-November/077281.html

If the simulation is performed a sufficiently long time, there would be a moment of saturation and we have got the sigmoid. According to our study of computer worms it is important to realize that this behavior is very similar to the behavior of biological viruses, and we can apply their models of propagation.

2.2 Stealthy worm

Using P2P technology introduces many benefits, it is much harder to identify and shut down a botnet with no central point of the network and to detect the bot master, because it can be represented by any node in the network. On the other hand (it was the case of Storm), there must be some kind of a central authority since not all nodes can be considered to be trusted. This problem is solvable with asymmetric cryptography and digital signatures.

The Storm botnet was the first one that began to use strong cryptographic ciphers like RSA. His successor Waledac eliminated many shortcomings, but contains implementation weaknesses that makes possible to switch off both these botnets completely. The digital signature guarantee is the most critical part of the worm design. Each worm should contain the public key of its owner that is necessary to verify the authenticity of instructions.

We recommend to use RSA key with the size at least 2 048 bits for this purpose. The botmaster in this case can estimate the list of authorized people who can carry out commands using the trusted central authority. In this case the worm validates clients' certificates.

2.2.1 Worm's upgrade

The worm's development and the intervention of anti-virus companies often causes the arms race. As in the case of very sophisticated worm Conficker, there is usually the fast reaction from the anti-virus/IDS/IPS companies (generated domain isolation architecture, AV signatures), on the other hand worms developers added the P2P functionality and extended the domain generation architecture. It is necessary to compile and distribute binaries for each version as well as to include the appropriate interpreter for scripting language to the worm's body.

Thanks to the scripting language you can easily add the possibility to download exploits for the worm's exploiting functionality or commands for the Command and Control functionality. The suitable candidate is the fast LUA interpreter which has only about 200 kilobytes. It is also used for the security scanner nmap with scripting support (NMAP scripting engine). At this time there is almost no information available about the existence of the worm with scripting language support, one of them is Worm.Win32.Leave[4] that uses it for downloading new plugins.

[4] http://www.virus-encyclopedia.com/virus/InternetWorms/worms148.html

2.2.2 Social network sites such as Command and Control

At the time of writing this article we are not aware of any practical worm's implementation that uses social networks to communicate. We have found only one example[5] of botnet that uses twitter for URLs exchanging, but with no obfuscation or cryptography.

We have proposed a solution that can be used with almost any type of social networking.

1. Worm's instructions can be stored to publicly available Twitter account with predetermined name. We suppose this name will be revealed and publicly known after a short period by AV companies.

2. The botmaster publishes his instructions including the computed digital signature and valid timestamp:

3. The worm's instance will check these instructions every few minutes and verify their timestamp. If these instructions are still valid, it will check the digital signature.

4. If the digital signature is valid, the worm will execute the instructions otherwise it will skip to the next step

5. The worm reveals that the twitter (or some other) account has been detected and blocked. The account name is hashed by strong hash function and the output is consequently converted into a set of allowable characters with cardinality n (number of displayable characters) that are used as a new user nickname during the registration. The worm will try to follow all other instructions from this hashed account.

Using this method the attacker can generate an indefinite number of possible accounts that can be used for downloading of new worm's instructions. Due to the fact that the indefinite number of recursive hashed functions is used, it will be difficult to block all these accounts or stop the existing social network.

2.2.3 Antivirus bypass with Metasploit Framework

Metasploit Framework is a tool for developing and executing exploit code for gaining full access over remote target machines [5]. The current version contains many encoders that can be used as a prevention against the recognition by most antivirus products. How this process looks exactly we can find in example the [6]. Most antivirus protections are already coverted by standard templates, so we can modify them or to create our owns. Unfortunately they are not publicly known,

[5] http://asert.arbornetworks.com/2009/08/twitter-based-botnet-command-channel/

because this information would help to develop new dangerous virus signatures. The list of current Metasploit Framework's encoders:

```
Framework Encoders
==================

Name                    Rank       Description
----                    ----       -----------
cmd/generic_sh          good       Generic Shell Variable ..
cmd/ifs                 low        Generic ${IFS} Substitution ..
cmd/printf_php_mq       manual     printf(1) via PHP magic_ ..
generic/none            normal     The "none" Encoder
mipsbe/longxor          normal     XOR Encoder
mipsle/longxor          normal     XOR Encoder
php/base64              great      PHP Base64 encoder
ppc/longxor             normal     PPC LongXOR Encoder
ppc/longxor_tag         normal     PPC LongXOR Encoder
sparc/longxor_tag       normal     SPARC DWORD XOR Encoder
x64/xor                 normal     XOR Encoder
x86/alpha_mixed         low        Alpha2 Alphanumeric Mixed ..
x86/alpha_upper         low        Alpha2 Alphanumeric Upper ..
x86/avoid_utf8_tolower  manual     Avoid UTF8/tolower
x86/call4_dword_xor     normal     Call+4 Dword XOR Encoder
x86/context_cpuid       manual     CPUID-based Context Keyed ..
x86/context_stat        manual     stat(2)-based Context Keyed ..
x86/context_time        manual     time(2)-based Context Keyed ..
x86/countdown           normal     Single-byte XOR Countdown ..
x86/fnstenv_mov         normal     Variable-length Fnstenv/mov ..
x86/jmp_call_additive   normal     Jump/Call XOR Additive ..
x86/nonalpha            low        Non-Alpha Encoder
x86/nonupper            low        Non-Upper Encoder
x86/shikata_ga_nai      excellent  Polymorphic XOR Additive  ..
x86/single_static_bit   manual     Single Static Bit
x86/unicode_mixed       manual     Alpha2 Alphanumeric Unicode ..
x86/unicode_upper       manual     Alpha2 Alphanumeric Unicode ..
```

2.2.4 Code armoring

In [7] it is presented an algorithm that makes it very difficult or in several cases almost impossible any reverse engineering and disassembling. The principle consists in the derivation key, which is dependent on the environment in which our binary is located. Malware analysts in this case can not detect the environment

on which to run the binary nor the executable code until completion.

- In the first step we need to generate a key that is derived from the environment and it does not change during the running period of the executable code, such as domain name or the name of the logged on user, which is the target of our attack.
- Unlike conventional methods, dynamically generated malware is not necessary on the disk (this should be true even in encrypted form), while it is prepared to run.
- However, disk may contain any other application where instructions are "transformed" in our malware. We can verify its presence using the checksum.
- This need has not to be tied to a specific architecture - only the specific string has to be used to represent our key.

This technique works by combining of multiple keys (username or domain) and the salts, the output itself will be the hash. This hash function provides a string of hexadecimal characters that are used for a substring that already represents the required instructions for malware. In order to get the desired output we have to generate salt appropriately. This section is the most time-consuming, but it is performed only once and we can distribute the calculation over several computers (botnet). This is useful mostly in the case that we have implemented server-side polymorphism and our worm changes itself every few minutes.

In the case when a wrong key is inserted into the hash function, we get a binary that is not executable and crashes with exception. We can store the checksum, but this information can serve as an useful information to the reverse engineering. For the deeper analysis it is possible to recover the decryption routine, salt, index of substrings and also an algorithm (domain) which represents the key. But it is no other way to determine the actual keys without bruteforcing and the victim becomes aware of malware functionality only after its launch. Currently, using heuristics and common detection techniques it is possible to detect this specific malware.

3 Summary

As we can see the current technologies make possible many improvements of computer worms:

- Usage of distributed P2P architecture instead of centralized IRC botnets increased the worm's potential damage.

- Strong cryptographic ciphers significantly minimize the worm's exposure.
- Worms can be dynamically reprogrammed using the scripting language in that way that their intentions remain unknown until their execution

References

[1] Joan Calvet, Carlton R. Davis and Pierre-Marc Bureau: *Malware Authors Don't Learn, and That's Good!*, 4th International Conference on Malicious and Unwanted Software (Malware'09), October 2009.

[2] Sinclair, G., Nunnery, C., Kang, B. B.-H.: *The Waledac Protocol: The How and Why*, iDefense, Univ. of North Carolina at Charlotte, Charlotte, NC, USA, October 2009.

[3] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran: *An Analysis of Conficker's Logic and Rendezvous Points*, http://mtc.sri.com/Conficker/, March 2009.

[4] Dave Piscitello: *Conficker Summary and Review*, https://www.icann.org/en/security/, May 2010.

[5] http://en.wikipedia.org/wiki/Metasploit_Project

[6] http://www.offensive-security.com/metasploit-unleashed

[7] John Aycock, Rennie deGraaf and Michael Jacobson: *Anti-disassembly using Cryptographic Hash Functions*, Springer-Verlag France 2006, May 2006.

[8] Ertug Karamatli: *Modern Botnets: A Survey and Future Directions*, Bogazici University, Turkey, January 2011.

Sledování uživatelů prostřednictvím webových technologií

Jaromír Dobiáš

E-mail: jaromir.dobias@mail.muni.cz

Abstrakt

Webové technologie pronikají čím dál víc do širokého spektra aplikací využívaných v našem každodenním životě. Kromě výhod, které s sebou přináší, jsou však také zdrojem mnohých hrozeb s dopadem na soukromí uživatelů. Programátoři vyvíjející aplikace na bázi webových technologií častokrát netuší, jaké může mít ta či ona komponenta použita v jejich aplikaci vliv na soukromí uživatelů. I samotní uživatelé využívající webové aplikace či konvenční aplikace obsahující webové prvky mají jen zřídka představu, jak a kým mohou být prostřednictvím těchto aplikací sledováni. Tento článek se proto zabývá vybranými webovými sledovacími prostředky a analyzuje jejich vlastnosti s ohledem na využití pro sledování uživatelů.

1 Úvod

Webové technologie zaznamenaly v posledním desetiletí výrazný pokrok. Z technologií, které kdysi umožňovaly pouze vytváření statického obsahu a jeho prezentaci online se postupně vyvinul silný nástroj, označován jako Web 2.0, s jehož pomocí lze dnes vytvářet plnohodnotné aplikace a programy. Ten se stal základním stavebním kamenem nepřeberného množství aplikací jako například e-shopů, internetových platebních systémů či sociálních sítí, které dnes využívají milióny lidí. Lidé však využíváním aplikací na bázi webových technologií také propagují do online světa mnoho informací o sobě samotných. Buďto přímo tím, že je vědomě poskytují konkrétní aplikaci, či nepřímo – vykonáním určité akce (např. stiskem tlačítka „tento obsah se mi líbí“ nebo pouhým prohlížením určitého obsahu dostupného online).

Webové technologie lze ale také velmi dobře využít právě k získávání informací o uživatelích a k jejich sledování. Navíc, díky čím dál tím větší propojenosti webových systémů (např. tzv. *mashups*) a možnostem spojitelnosti uživatelských

akcí (např. prostřednictvím *single sign-on*) napříč jednotlivými systémy, lze často spojit dílčí parciální identity[1] uživatele (v rámci daných systémů), či dokonce svázat provedené akce s reálnou identitou uživatele. Jinými slovy, je možné zjistit, že různé uživatelské profily s neidentickými identifikátory (např. s různými přihlašovacími jmény či avatary) v různých systémech patří jedné osobě nebo dokonce odhalit její reálné jméno či fyzickou lokaci.

Agregací a analýzou těchto dat lze sestavit obraz o uživatelích, jejich preferencích, zájmech a názorech, pohybu (nejen na Internetu, ale mnohdy i v reálném světě) a odhalit mnoho dalších potenciálně citlivých aspektů.

Relativně malé povědomí o existujících možnostech sledování uživatelů prostřednictvím webových technologií a čím dál větší sofistikovanost sledovacích prostředků objevených v uplynulých letech byly hlavní motivací pro vznik tohoto článku. Článek se snaží poskytnout přehled současných webových prostředků umožňujících sledování uživatelů prostřednictvím webových technologií. Popisuje principy fungování daných prostředků a demonstruje, k čemu a jak mohou být použity.

2 Analýza charakteristických aspektů webových sledovacích prostředků

S rozšiřujícími se možnostmi a rostoucí komplexností webových technologií roste i množství způsobů jejich využití pro účely sledování uživatelů. Sledovací prostředky na bázi webových technologií se od sebe vzájemně liší v různých aspektech např. podle toho, kdo má možnost prostřednictvím daného prostředku jak, zjistit jaké informace a o kom. Analýze uvedených aspektů je věnována tato kapitola.

2.1 Typ pozorovatele

Typ pozorovatele sledovacího prostředku určuje, kdo je schopen využít daný prostředek na sledování uživatelů. Možnost využití prostředku vyplývá z jeho technické podstaty. Pozorovatelem se zde rozumí entita, která využívá sledovací prostředek k získávaní informací o uživatelích. Podle charakteristického způsobu nasazení sledovacího prostředku rozlišujeme následující typy potenciálních pozorovatelů:

Provozovatel online služby – jedná se o nejobvyklejší případ, kdy samotný provozovatel sleduje uživatele své služby. Z uvedených má tento typ te-

[1] Identita individuálního jedince může být tvořena mnoha *parciálními identitami*, z nichž každá reprezentuje daného jedince ve specifickém kontextu nebo roli [2]. Typickým identifikátorem parciální identity je pseudonym.

oreticky největší potenciál získat co největšího množství informací o sledovaných uživatelích. Je to z toho důvodu, že provozovatel má obvykle plnou kontrolu nad službou, kterou provozuje a není tím pádem technicky omezen ve výběru nasaditelných sledovacích prostředků.

Poskytovatel vzdáleného obsahu – sledování v tomto případě vykonává poskytovatel obsahu umístěného na vzdáleném serveru pod jeho kontrolou[2]. Sledováni jsou uživatelé využívající obsah nebo službu, do které je vzdálený obsah vložen. Pro tento typ sledování je podstatné, že vkládání vzdáleného obsahu je realizováno ve smyslu funkce `embed`. To znamená, že do sledované služby resp. obsahu se vloží pouze ukazatel odkazující na reálné umístění vzdáleného obsahu. Možnost sledování je pak zprostředkována tím, že při manipulaci se sledovaným obsahem (např. návštěva sledované stránky nebo otevření sledovaného dokumentu) se zároveň načítá i vzdálený obsah pod kontrolou poskytovatele. Poskytovatel vzdáleného obsahu tak má prostředek pro sledování uživatelů. Poskytovatelem vzdáleného obsahu, který má potenciální prostředky pro sledování uživatelů může být například poskytovatel vkládaného multimediálního obsahu, poskytovatel vkládaných reklamních ploch nebo poskytovatel analytických služeb jakou je návštěvnost webu apod.

Interní uživatel služby – sledování uživatelů v tomto případě realizuje interní uživatel služby. Ten v souladu se svými pravomocemi využívá možnosti, které mu služba poskytuje k nasazení sledovacích prostředků. Pro tento typ je charakteristické, že spektrum použitelných sledovacích prostředků závisí na tom, jaký obsah služba umožňuje internímu uživateli v rámci jeho pravomocí přidávat a kam. Příkladem tohoto typu může být např. uživatel přidávající do diskusního fóra příspěvek s obrázkem načítaným ze vzdáleného umístění nebo správce obsahu používající svůj vlastní soubor s definicí kaskádových stylů pro konfiguraci vizuální podoby obsahu.

Externí uživatel služby – pro tento typ je charakteristické, že sledování realizuje externí uživatel služby využívané pro účely sledování. Na rozdíl od interního uživatele, není externí uživatel vůči službě autentizován resp. nepotřebuje žádné specifické oprávnění k tomu, aby mohl službu využívat. Typickým příkladem jsou např. webové služby, které umožňují přidávat upravovat či jinak modifikovat obsah služby neautentizovaným uživatelům. Jiným příkladem mohou být multimediální soubory obsahující vzdálený obsah či skripty.

[2]Předpokládáme, že poskytovatel vzdáleného obsahu má na serveru, kde je obsah hostování dostatečnou míru oprávnění a prostředků jako např. možnost vytváření a publikování PHP skriptů apod.

2.2 Rozsah získatelných dat a informací

Z pohledu rozlišování charakteristických vlastností sledovacích prostředků je potřeba rozlišovat mezi *rozsahem získatelných dat* a *rozsahem získatelných informací*. V prvním případě se jedná o objektivní faktor nezávislý na pozorovateli, zatímco v druhém případě rozsah závisí na pozorovateli. Pokud je pozorovatel schopen ze získaných dat vytěžit informaci, která je předmětem jeho zájmu, hovoříme že sledovací prostředek poskytuje pozorovateli užitečnou informaci.

Rozsah získatelných informací určuje, co vše lze o sledovaných uživatelích prostřednictvím určitého sledovacího prostředku zjistit, tedy jaký typ informací o uživatelích je prostřednictvím daného prostředku možné získat. Je obtížné striktně vymezit typy informací získatelných prostřednictvím webových sledovacích prostředků. A to z toho důvodu, že rozsah těchto informací závisí kromě použitého sledovacího prostředku i na mnoha dalších faktorech. Tím je např. prostředí služby do něhož je sledovací prostředek nasazen, množství a typ uživatelů využívajících danou službu či frekvence využívání.

Důležitým faktem je, že data získaná prostřednictvím webových sledovacích prostředků vyhodnocuje každý pozorovatel individuálně na základě své subjektivní znalosti. Ze stejného vzorku získaných dat tedy obvykle různí pozorovatelé získají různé informace.

Představme si například, že pozorovatel P_1 zasílá firmě X elektronickou žádost o pracovní místo k níž přikládá svůj životopis. Před odesláním však do životopisu vloží také sledovací mechanizmus odhalující IP adresu zařízení, na němž je dokument prohlížen a aktuální čas. Kdykoliv někdo daný dokument otevře, zaznamená se pozorovateli P_1 IP adresa příslušného zařízení a čas, kdy k otevření dokumentu došlo. Řekněme, že pozorovatel prostřednictvím sledovacího mechanizmu získal 3 záznamy v časovém rozestupu dvou dnů, přičemž třetí záznam pochází z jiné IP adresy, než první dva z předchozího dne, avšak všechny jsou z adresního rozsahu firmy X. Pozorovatel P_1 může na základě těchto dat usuzovat, že jeho životopis firmu X zaujal a postoupil do vyššího kola. Řekněme, že na podnět systému varování před nepracovními aktivitami pracovníků v pracovní době se k záznamům pozorovatele P_1 dostal také administrátor firmy X (pozorovatel P_2). Ten je schopen na základě IP adres a znalosti vnitřní firemní infrastruktury dohledat jména pracovníků, kterým tyto adresy patří. Bez další znalosti, kterou disponuje pozorovatel P_1 či bez další analýzy není spozorovatel P_2 schopen ze stejných dat vyvodit stejnou informaci, jako pozorovatel P_1.

2.3 Ideální model sledovaného uživatele

Ideální model sledovaného uživatele charakterizuje typ uživatele, při němž nasazení určitého sledovacího prostředku poskytuje pozorovateli největší přínos. U sledovacích prostředků popisovaných v tomto článku možno rozlišovat 3 modely sledovaného uživatele, a to:

Skupina uživatelů se společným zájmem – nasazený sledovací prostředek poskytuje pozorovateli největší přínos, pokud má cílová skupina určitý společný zájem, kvůli němuž využívá elektronický zdroj monitorovatelný pozorovatelem. Příkladem může být například skupina uživatelů využívajících diskuzní fórum k výměně zkušeností stýkajících se určitého zboží. Stejně tak však může být předmětem zájmů skupiny třeba výměna informací o výbušných systémech či výrobě a prodeji omamných látek.

Jednotlivec označitelný unikátním identifikátorem – nasazený prostředek je v tomto případě nejpřínosnější pro pozorovatele, pokud mu dává možnost naak co nejdelší možnou dobu označit uživatele v rámci určité skupiny dostatečně unikátním identifikátorem tak, aby následně byl schopen označeného uživatele jednoznačně odlišit od ostatních uživatelů. Příkladem jsou např. uživatelé, kterým služba vytvoří cookie(podrobně probráno v další kapitole).

Jednotlivec s výrazně odlišnými charakteristickými atributy – nasazený prostředek poskytuje pozorovateli největší užitek, pokud se jednotlivec liší od ostatních jedinců v rámci sledované skupiny tím, že má jeden či více atributů, které nemají ostatní jedinci nebo hodnoty jednoho či více atributů má výrazně odlišné ve srovnání s ostatními. V praxi to může být například uživatel, který má nainstalovaný velmi specifický typ prohlížeče ve srovnání s ostatními uživateli zachycenými sledovacím prostředkem.

2.4 Přesnost získaných informací

Přesnost získaných informací určuje, jak přesné údaje je schopen pozorovatel na základě použitého sledovacího prostředku získat. Přesnost můžeme chápat ve dvou rozměrech a to jako (1) pravděpodobnost, že informace odvozená na základě dat získaných prostřednictvím sledovacího prostředku je pravdivá nebo (2) pravděpodobnost, že vazba mezi odvozenou informací a identitou jedince je pravdivá. Obecně lze říci, že čím větší sledovaný prostor je pozorovatel schopen pokrýt čím delší dobu, tím přesnější informace o sledovaných uživatelích je potenciálně schopen získat. Přesnost získaných informací je přitom podmíněna schopností pozorovatele detekovat, že se určitá událost stala s vysokou mírou pravděpodobnosti a schopností svázat tuto událost s konkrétním identifikovatelným[3] subjektem.

2.5 Časový rámec sledování

Časový rámec sledování je doba začínající nasazením sledovacího prostředku a končící trvalým přerušením sledování. Nasazení sledovacího prostředku provádí

[3]Identifikovatelnost zde není chápána jako absolutní veličina, ale jako pravděpodobnostní parametr.

obvykle pozorovatel nebo jiná entita, spolupracující s pozorovatelem. Přerušení sledování může provést nejen pozorovatel či spolupracující entita, ale také sledovaný uživatel resp. skupina uživatelů nebo může být vyvoláno sledem událostí v prostředí nasazeného sledovacího prostředku. Při přerušení sledování rozlišujeme případ, kdy je sledovací prostředek (pokud existuje) (1) ponechán ve sledovaném prostředí nebo (2) odstraněn. V rozmezí časového rámce sledování lze ještě rozlišovat časové úseky, v nichž je sledování efektivní. Efektivní sledování je stav, kdy nasazený sledovací prostředek poskytuje pozorovateli užitečnou informaci. V ideálním případě časové rozpětí efektivního sledování koresponduje s časovým rámcem sledování.

Pokud nasazený sledovací prostředek přestane poskytovat pozorovateli užitečnou informaci a není pravděpodobné, že by se tento stav v budoucnu změnil, pozorovatel obvykle přerušuje sledování. Důvodem přerušení může být také to, že sledovací prostředek poskytl pozorovateli požadovanou informaci a splnil tím účel sledování.

2.6 Prostorový rámec sledování

Prostorový rámec sledování charakterizuje prostor, v němž je možné vykonávat sledování. V elektronickém světě nelze pohlížet na prostor tak, jak jej chápeme z analogového světa. Prostorem v tomto smyslu proto rozumíme množinu zdrojů digitálního světa schopných interagovat s webovým prostředím na bázi webových technologií. Zjednodušeně řečeno vše, co je schopno při své interpretaci vyvolat HTTP dotaz webovému serveru lze považovat za součást tohoto prostoru. Prostorový rámec sledování pak určuje množství a charakter zdrojů, které je při nasazení určitého sledovacího prostředku možné sledovat. O zdrojích aktivně monitorovaných pozorovatelem říkáme, že jsou v jeho zorném poli. Čím více zdrojů pozorovatel aktivně monitoruje, tím širší je jeho zorné pole a tím více informací je schopen potenciálně získat.

2.7 Detekovatelnost sledovacího prostředku

Detekovatelnost sledovacího prostředku charakterizuje schopnost resp. možnost sledovaných uživatelů detekovat, že se sledovací prostředek nachází v daném prostředí. Dalším aspektem je schopnost detekovat, že daný sledovací prostředek je aktivní, příp. kdo nebo co je předmětem sledování. Detekovatelnost sledovacího prostředku určují faktory jako např. existence a dostupnost automatizovaných nástrojů pro jejich odhalení, expertní znalost sledovaného subjektu či skupiny či technologická podstata sledovacího prostředku. Za nejméně nápadné lze považovat prostředky, které nároky na přidávání dodatečných sledovacích elementů nad rámec přirozené funkčnosti sledovaného prostředí jsou minimální. Čím více

splývá sledovací prostředek s přirozeným návrhem prostředí a jeho funkčních vlastností, tím méně je nápadný.

3 Webové prostředky pro sledování uživatelů

Předchozí kapitola se věnovala výčtu a popisu charakteristických vlastností, které lze rozlišovat u webových sledovacích prostředků. Tato kapitola se zabývá již přímo jednotlivými sledovacími prostředky v kontextu uvedených vlastností.

3.1 Sledování prostřednictvím HTTP Cookie

HTTP Cookie (dálen jen cookie) je nástroj protokolu HTTP umožňující webovému serveru vložit stavovou informace do klientského prohlížeče. Klientský prohlížeč, jemuž webový server nastaví cookie, pak při další komunikaci s tímto serverem zpětně reflektuje serveru stavovou informaci z cookie prostřednictvím HTTP požadavku.

Nástroj cookie lze využít k vložení unikátního identifikátoru do klientského prohlížeče. Prohlížeč pak při následné komunikaci se serverem, který vložení cookie realizoval, automaticky sděluje hodnotu vloženého identifikátoru a zajišťuje mu tím informaci, o kterého uživatele s nímž v minulosti již přišel do styku se jedná. Mechanizmus cookie umožňuje serveru jednoznačně určit uživatele, s nímž opakovaně komunikoval v rámci skupiny uživatelů, kteří mají v daném čase přidělen serverem také svůj specifický identifikátor. Díky možnosti využívat cookie pro účely sledování uživatelů lze tento webový nástroj považovat za sledovací prostředek.

3.1.1 Typ pozorovatele

Sledovacímu prostředku cookie nejvíce odpovídá *provozovatel online služby* jako charakteristický *typ pozorovatele* (viz 2.1). To znamená, že tento sledovací prostředek se nejlépe hodí pro případy, kdy je v roli pozorovatele provozovatel online služby. Jak však uvidíme dále (viz 3.3), mechanizmus cookie je využíván i dalším sledovacím prostředkem s širším spektrem typů pozorovatelů.

3.1.2 Rozsah získatelných dat a informací

Co se týče *rozsahu získatelných dat*, cookie poskytuje serveru zpětně pouze data, která server sám vložil do klientova prohlížeče. Na základě těchto dat je pak možné svázat akce uživatele vykonané v zorném poli pozorovatele a realizovat tak např. profilování. Pro pozorovatele není podstatné, že sledovaný uživatel je reprezentován pouze na první pohled nic neříkajícím číslem. Pro pozorovatele je

plně dostačující záruka, že dané číslo patří jedné entitě a informace, že uživatel s tímto číslem např. denně navštěvuje obchodní portál pozorovatele, kde se zajímá především o nákup výpočetní techniky společnosti X a průměrně utratí nákupem na portálu průměrně 1 337 Kč měsíčně.

3.1.3 Ideální model sledovaného uživatele

Sledovací prostředek cookie nejvíce odpovídá typu *jednotlivce označitelného unikátním identifikátorem*. Znamená to, že prostředek cookie je vhodně aplikovatelný v případě, kdy předmětem sledovaného zájmu je jednotlivec. Identifikace jednotlivce v rámci sledovaných subjektů je v zorném poli pozorovatele realizována na bázi unikátního identifikátoru.

3.1.4 Přesnost získaných informací

Cookie poskytuje pozorovateli informaci, že akce sledovaného uživatele je svázána s jeho identitou reprezentovanou identifikátorem. Cookie však poskytuje pouze informaci o identitě sledovaného uživatele, avšak nikoliv o samotné akci vykonané daným uživatelem. Sledovací prostředek cookie by se tak v tomto smyslu mohl jevit jako velmi přesný, jelikož pravděpodobnost pravdivosti informace o identitě uživatele je za ideálních okolností velmi vysoká. Problémem však je, že identifikátor může uživatel z prohlížeče velmi snadno odstranit. Pokud k tomu dojde, server při nejbližší další komunikaci nastaví uživateli nový identifikátor. Tím se však již daný uživatel jeví z pohledu pozorovatele jako jiná entita a ztrácí se tím pádem sjednocující element. Jednoduše řečeno, ze dvou odloučených zdrojů popisujících aktivitu určitého uživatele lze obvykle získat méně užitečné informace, než když se tyto zdroje sloučí. Je proto potřeba myslet na to, že přesnost získaných informací na základě sledovacího prostředku cookie je redukována snadnou možností rozbití identity uživatele na dílčí subkomponenty mezi, kterými neexistuje sjednocující element.

3.1.5 Časový rámec sledování

Z časového pohledu je sledování na základě cookie iniciováno v momentě, kdy pozorovatel v rámci své služby zahájí identifikaci uživatelů na základě jednoznačného identifikátoru v cookie (pro účely sledování[4]). Sledování na základě cookies je nejefektivnější, když je akce každého uživatele v zorném poli pozorovatele svázaná s původním identifikátorem obdrženým při prvním kontaktu uživatele se službou provozující sledování až do přerušení sledování. Efektivitu

[4]Cookie bývá obvykle nasazována pro účely autentizace. Pokud však primárním cílem nasazení není přihlášení uživatele, ale jeho sledování, pak takové použití mechanizmu cookie chápeme jako sledovací prostředek.

sledování tohoto nástroje minimalizuje odstraňování identifikátoru před každým novým dotazem službě v zorném poli pozorovatele. Sledování je přerušeno odstavením vkládání nových unikátních identifikátorů do klientských prohlížečů komunikujících se službami v zorném poli pozorovatele. Sledovací efekt však může přetrvávat i po přerušení sledování, protože cookie zůstávají uloženy na straně klienta buď do vypršení jejich platnosti (pokud je shora omezena), nebo do doby jejich explicitního odstranění.

3.1.6 Prostorový rámec sledování

Prostorový rámec se v případě cookie zaměřuje primárně na vlastní infrastrukturu pozorovatele. Pozorovatel využívající prostředku cookie je schopen sledovat aktivity uživatelů využívajících jeho služeb.

3.1.7 Detekovatelnost sledovacího prostředku

Současné prohlížeče jsou schopné nejen oznamovat či dotazovat se uživatelů na přijetí či zamítnutí cookie, ale nabízí i možnosti na jejich automatického mazání. Kromě toho moderní prohlížeče nabízí také možnost zobrazit všechny cookie uložené v prohlížeči. Obtížnost detekovatelnosti cookies je tedy prakticky nulová.

3.2 Sledování prostřednictvím Local Shared Object

Local Shared Object[5] (dále jen LSO) je poměrně mladá technologie využívající podobný mechanizmus, jako cookie pro uložení stavové informace na straně klienta. Na rozdíl od cookie, kde se stavová informace přenáší mezi klientským prohlížečem a serverem prostřednictvím HTTP hlaviček, LSO k přenosu stavové informace využívá multimediální soubory typu *SWF* interpretované pomocí *Adobe Flash Playeru*[6].

Velmi zásadní rozdíl LSO oproti cookie je v ukládání stavové informace. Zatímco stavová informace obsažená v cookie se ukládá do klientského prohlížeče, v případě LSO se ukládá lokálně na disk do adresářové struktury Adobe Flashe. V důsledku toho je každá stavová informace LSO sdílena mezi všemi prohlížeči daného uživatele.

Představme si např. hlasovací anketu, která se snaží zabránit tomu, aby stejný uživatel hlasoval víc, než jednou. Při klasickém způsobu ochrany na základě IP adresy a cookie by stačilo, kdyby uživatel smazal cookie obdrženou od serveru a využil by při druhém hlasování připojení přes nějaký anonymní proxy server. Pokud by však ochranný mechanizmus používal také prostředek

[5] Označován také jako *Flash Cookie*.

[6] http://www.adobe.com/products/flashplayer/

LSO, uživateli by nebyl schopen opětovného hlasování dosáhnout ani pokud by se ochranný mechanizmus obejít použitím jiného prohlížeče. LSO se často používá v kombinaci s jinými sledovacími nástroji pro zachování spojitelnosti akcí uživatele z pro pozorovatele.

3.3 Sledování prostřednictvím webové štěnice

„Webová štěnice“ neboli web bug je další typ webového sledovacího prostředku. Pro tento prostředek je typické, že využívá možnost vkládání vzdáleného obsahu do obsahu, který má být sledován. Pro účely sledování zde není potřebné rozlišovat, jaký datový obsah je ve vzdáleném obsahu obsažen, ale pouze to, že se načítá ze vzdáleného umístění obdobně, jako to realizuje funkce `embed` (typicky obrázek apod.). Sledovací efekt je pak zprostředkován tím, že kdykoliv je sledovaný obsah (obvykle na webové stránce) někým zobrazen, tímto zobrazením zároveň aktivuje načtení vzdáleného obsahu [1].

Poskytovatel vzdáleného obsahu je pak schopen na základě komunikační režie zaznamenané z požadavku na vzdálený obsah odvodit určitou informaci, která dává do vztahu sledovaný obsah a charakterizuje uživatele, který sledovaný obsah zobrazil (resp. interpretoval). Kromě toho web bug umožňuje také ještě daného uživatele „označkovat“ prostřednictvím cookie tím, že spolu se vzdáleným obsahem posílá také cookie. To má za následek, že kdykoliv daný konkrétní uživatel znovu navštíví sledovaný obsah, bude mít poskytovatel obsahu také informaci o unikátní identitě sledovaného uživatele. To se typicky využívá např. v e-shopech pro marketingové účely, jelikož tak lze sledovat kolikrát si např. určitý uživatel prohlížel stránku určitého produktu apod. Kromě lze prostřednictvím web bugu získat i jiné identifikátory jako např. IP adresu, čas nebo HTTP hlavičku `referer`, které mohou prozradit z jaké lokality a kdy s sledovaný uživatel sledovaný obsah prohlížel a případně na jaké stránce se před přístupem k sledovanému obsahu nacházel.

3.4 Sledování na základě krádeže historie navštívených URL

Technika krádeže historie navštívených URL funguje díky tomu, že většina moderních prohlížečů si zaznamenává historii navštívených odkazů a při jejich zobrazování na stránce zobrazují navštívené odkazy jinou barvou než nenavštívené. Tuto vlastnost využívá pozorovatel tak, že si zkonstruuje (typicky v JavaScriptu) jednoduchý kód procházející položkami seznamu předdefinovaných odkazů a dotazuje se, které položky ze seznamu se vykreslují barvou nenavštíveného odkazu. Po průchodu všemi položkami lze takto separovat navštívené odkazy od nenavštívených a poslat informaci o navštívených odkazech pozorovateli. Podmínkou je, že kód se musí spustit v prohlížeči sledovaného uživatele.

Informace, kterou je pozorovatel schopen získat závisí na tom, jaké odkazy naplní do testovacího seznamu. Prostřednictvím této techniky lze kromě navštívených odkazů detekovat také parametry zasílané v URL HTTP metodou GET. Na základě této vlastnosti je pak možné zjišťovat například jaké vyhledávané výrazy uživatel zadával do vyhledávacích služeb, které přenášejí vyhledávané výrazy metodou GET.

3.5 Sledování na základě kontextových parametrů prohlížeče

Schopnost určit identitu uživatele nemusí být závislá pouze na unikátnosti jednoznačného identifikátoru. Uživatelé používají prohlížeče, které se vzájemně liší v mnoha atributech, a jejichž jsou měřitelné pozorovatelem. Tyto atributy vytváří jakýsi otisk, který do identifikuje prohlížeč. Schopnost identifikovat uživatele je pak možné realizovat na základě tohoto otisku, pokud je dostatečně odlišný od otisků dalších sledovaných uživatelů. Obvyklými parametry na základě nichž je možné identifikaci provádět jsou typicky, typ prohlížeče, verze, lokalizace, nainstalovaná písma, nainstalované pluginy, rozlišení obrazovky a mnoho dalších.

4 Závěr

Webové technologie se rapidně rozvíjejí a poskytují čím dál tím propracovanější nástroje a prostředí pro vznik pokročilých aplikací. Některé nástroje však často mají vedlejší efekty umožňující sledování uživatelů a možno je proto považovat za sledovací prostředky. U těchto prostředků je možné rozlišovat určité charakteristické vlastnosti zejména podle toho, kdo je schopen s využitím určitého prostředku, zjistit jaké informace a o kom. Smyslem tohoto článku bylo ukázat jak jednotlivé sledovací prostředky fungují, v jakých případech a k čemu je lze použít. Článek poskytl analýzu charakteristických vlastností sledovacích technik pro lepší pochopení, co lze od nich očekávat při jejich nasazení na sledování uživatelů. V tomto článku rozhodně nebyly prezentovány všechny existující techniky a prostředky webových technologií umožňující sledování uživatelů. To však z praktického hlediska není ani možné, protože nové a nové techniky se pořád objevují, jakožto i způsoby jejich nových aplikací. Dalším důvodem je, že vzájemnou kombinací sledovacích technik lze získat velmi specifické sledovací vlastnosti, s kterými se lze setkat pouze ve výjimečnějších případech. Cílem tohoto článku bylo však poskytnou obecný náhled na tuto problematiku, aby bylo jasnější, jaká rizika číhají na uživatele v prostředí webu či na co si dávat pozor při psaní webových aplikací.

Literatura

[1] Dobias, J.: Privacy Effects of Web Bugs Amplified by Web 2.0, In *Privacy and Identity Management for Life*, Springer Boston, 2011.

[2] Pfitzmann, A., Hansen, M.: *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management.* Aug 2010, http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.bib, v0.34.

EGI Security challenge 5: Lehce na cvičišti...

Radoslav Bodó, Daniel Kouřil

E-mail: bodik@civ.zcu.cz, kouril@ics.muni.cz

Abstrakt

Evropská gridová infrastruktura (EGI) poskytuje přístup k rozsáhlým výpočetním a úložným kapacitám, které využívají uživatelé z Evropy i celého světa. Zajištění dostatečné úrovně bezpečnosti je přirozeně velmi důležitou součástí provozu celé infrastruktury. Je to ovšem zároveň úkol náročný na koordinaci a komunikaci, protože zahrnuje komunikaci s velkým počtem správců a bezpečnostních expertů, kteří zodpovídají za jednotlivé zdroje připojené do EGI.

Za oblast provozní bezpečnosti je zodpovědný tým EGI CSIRT, který mimo jiné koordinuje řešení bezpečnostních incidentů, které se v EGI objeví. Jednou z aktivit EGI CSIRT je také pořádání „cvičení", tzv. Security challenges, jejichž cílem je simulovat bezpečnostní problémy. Během řešení těchto incidentů se ověřuje, že informace jsou správně a včas předávány mezi zúčastněnými stranami a získané informace jsou kvalifikovaně zpracovávány.

V tomto příspěvku popisujeme poslední takové cvičení, které proběhlo v květnu 2011. Cvičení jsme se aktivně zúčastnili a podělíme se zejména o zkušenosti s forenzní analýzou, které mohou být užitečné i pro řešení reálných incidentů.

1 Úvod

Přestože odborným termínům v oblasti distribuovaného zpracování informací aktuálně vévodí *cloud computing*, nemůžeme pominout i jiná řešení, která nemají takový marketingový tah, nebo ne naopak přesunula z oblasti marketingu do zaběhnutých kolejí běžného života. Jednou takovou technologií jsou bezesporu gridy. Grid je distribuované prostředí, které nabízí přístup k velkým výpočetním a úložným kapacitám a zjednodušuje práci s nimi.

První gridová řešení se začala objevovat před více než deseti lety a od té doby vzniklo několik iniciativ poskytující gridové prostředí. K největším z nich patří Evropská gridová infrastruktura EGI[1]. EGI sdružuje tzv. národní gridové infrastruktury, které zastřešují poskytovatele zdrojů na úrovni jednoho státu.

[1] http://www.egi.eu/

V rámci ČR je takto zapojeno několik clusterů a diskových úložišť z několika univerzit, Akademie věd a sdružení CESNET. EGI pak zajišťuje globální koordinaci jednotlivých národních aktivit.

Uživatelská základna EGI přesahuje sto tisíc aktivních uživatelů, kteří měsíčně vyprodukují přes 15 milionů výpočetních úloh z nejrůznějších oblastí vědy a výzkumu. EGI např. pomáhá zpracovávat data z urychlovače částic, který je provozován v ženevském CERNu.

Při takto rozsáhlé infrastruktuře a počtech uživatelů je přirozené, že zajištění bezpečnosti celého prostředí je nesnadná úloha. O provozní bezpečnost se stará tým EGI CSIRT, který sleduje aktuální hrozby a případně vydává upozornění, pokud se objeví nějaký bezpečnostní problém, který by mohl mít dopad na provoz EGI. EGI CSIRT spoléhá do značné míry na bezpečnostní experty na nižších úrovních, zejména tam, kde je potřeba zajistit komunikaci přímo s koncovými poskytovateli zdrojů nebo uživateli. Při běžném řešení rozsáhlejšího incidentu se tak řeší komunikace mezi poměrně velkým počtem lidí, což je náročné na koordinaci. Pro řešení incidentů v EGI existují politiky i prováděcí směrnice, které specifikují povinnosti všech „hráčů“. Samotná existence těchto pravidel ale automaticky nezajistí, že komunikace vždy probíhá správně a včas. V případě incidentů, zejména těch rozsáhlejších, je potřeba vyrovnat se se zvýšeným tlakem a stresem, který je způsoben tlaky ze všech stran, často včetně managementu nebo médií.

Ve snaze připravit sebe i všechny další potenciální partnery pořádá EGI cvičení, která simulují napadení infrastruktury rozsáhlým útokem. Řešení takového útoku vyžaduje koordinaci řady správců i bezpečnostních expertů, včetně samotného EGI CSIRT a je tedy dobrou příležitostí, jak ověřit připravenost institucí a pracovníků na všech úrovních.

V příspěvku se zaměříme na poslední takové cvičení, tzv. *EGI security challenge*, které proběhlo v závěru května 2011. V rámci tohoto cvičení byly vytipovány výpočetní clustery ve většině zapojených národních iniciativách, na kterých byla běžným způsobem spuštěna speciálně připravená úloha. Tato úloha obsahovala programy simulující chování „pěšáka“ botnetu, tj. po spuštění se snažila přihlásit na řídící centrum a provádět získané povely.

2 Cvičení SSC5

Na přípravě cvičení SSC[2] pracoval tým EGI CSIRT několik měsíců a výsledkem je poměrně sofistikovaný systém pro správu botnetu a jeho vizualizaci. Cvičení probíhalo pod dozorem operátora cvičení, který hrál roli útočníka. Cvičení simulovalo situaci, kdy útočník získá oprávnění pro přístup k velké části infrastruk-

[2] https://wiki.egi.eu/wiki/EGI_CSIRT:Security_challenges

tury, tj. k spouštění dávkových úloh a k manipulaci s daty. Celá simulace zahrnovala čtyřicet míst z dvaceti států. Informace o cvičení byli předem účastníkům oznámeny, aby vědeli, že se jedná o simulaci.

Na počátku operátor vložil do dávkového systému úlohy pro všechny zúčastněné klustery. Po svém spuštění se úlohy přihlásí na řídící centrum botnetu (C&C) a jsou posléze připraveny vykonávat příkazy od centra. Řídící centrum i další pomocné komponenty byly opět správovány operátorem cvičení. Ke spouštění jednotlivých botů docházelo postupně, podle toho, kdy se uvolnilo místo na výpočetnách uzlech.

Po spuštení části botnetu byla rozeslána hlášení o podezřelých síťových aktivitách správcům dvěma zapojených klusterů. Tato hlášení se podobala výzvám, které občas posílají jiné CSIRT týmy v okamžiku, kdy detekují průnik nebo podezřelou aktivitu. Od tohoto okamžiku se koordinace incidentu ujal EGI CSIRT.

V rámci prvních analýz „napadených“ strojů byly objeveny dvě podezřelé IP adresy, se kterými jednotliví boti komunikovali. Po získání těchto adres jsme využili monitorovací systém na bázi netflow dat FTAS, který provozuje sdružení CESNET. Pomocí systému FTAS jsme následně objevili komunikaci, která proběhla s těmito IP adresy zevnitř sítě CESNET2 a měli jsme tedy důvodné podezření o „kompromitování“ strojů, za které jsme přímo zodpovědní a zahájili jsme tedy vyšetřování v rámci českého gridu.

Vzhledem k pravděpodobnému rozsahu incidentu jsme uspořádali videokonferenci, které se účasnili administrátoři a bezpečnostní experti dotčených institucí a dohodli další postup. Paralelně s těmito aktivitami probíhala čilá komunikace na úrovni EGI CSIRT a nová zjištění sdílena mezi všemi účastníky.

První analýza kompromitovaného klusteru odhalila konkrétní stroj, na kterém úloha běžela. Administrátoři dále izolovali vlastní program bota a provedli základní forenzní ohledání napadeného stroje. Získané soubory z disku byly dále podrobeny podrobnému ohledání, které je detailně popsáno v následujících kapitolách.

Průběh celého cvičení shrnují dva propagační snímky, které jsou dostupné z http://www.nikhef.nl/grid/ndpf/files/Global-Security-Exercises/FIRST2011/

3 Analýza malware egi.ssc5.wopr

Vzorek nalezené virové úlohy obsahoval:

- spouštěcí shell skript,
- 2 spustitelné soubory – malware `egi.ssc5.wopr`[3]
- Pakiti klienta[4].

[3] http://en.wikipedia.org/wiki/WarGames

[4] Pakiti: A Patching Status Monitoring Tool – http://pakiti.sourceforge.net/

Ve většině případů je doporučeno z napadeného stroje sejmout forenzní obrazy stavu operačního systému [6, 7]:

- spuštěné procesy – `ps faxu`
- otevřené komunikační kanály – `netstat -nlpa; ipcs`
- obraz RAM – `dd /dev/k?mem`
- obrazy disků a jejich kontrolní součty – `dd; sum`...

3.1 Úroveň 1: Sběr základních informací

Po zachycení byla provedena analýza chování spuštěním vzorku ve sledovaném prostředí testovacího výpočetního uzlu a ze 4 sekundového běhu byl pořízen záznam výstupních proudů `tmp.stdout` a `tmp.stderr`, záznam síťové aktivity `wopr.tcpdump` a záznam interakce s operačním systémem `strace.out`.

```
|-- [ 12K] pakiti-ssc-client              cee7c9cefa1e94fb0a3d2bb18189d85bebb01632
|-- [ 479] ratatosk.sh                    0aa241f5ca224ee54b6c3d1971736d9a6a5ad17d
|-- [1.6M]
   wopr_build_centos64.ANALY_GLASGOW      a5cd77ab855e274d201021f12f7f96fcc0c8f367
|-- [1.2M]
   wopr_build_v6_debian32.ANALY_GLASGOW   8f1d97d80afde2872dd18df6b74228fd9c0139ac
|-- [ 95K] strace.out                     a2ffa7beefafcfc6d18eacbd9fddbd7a40ca9f93
|-- [   0] tmp.stderr                     da39a3ee5e6b4b0d3255bfef95601890afd80709
|-- [ 88K] tmp.stdout                     d6159289dab635781f56fa8608860cdd01ea026d
`-- [331K] wopr.tcpdump                   4f502e8a8d8219d2b965e0e98f947c0e08217010
```

Obr. 1: Základní data zachyceného vzorku

Ze záznamu síťové aktivity (obr. 2) je možné zjistit, že malware používá HTTP C&C [15] (webové řídící centrum) na IP adrese 195.140.243.4. Po svém startu ohlásí centrále základní údaje o stanici na které je spuštěn (požadavky `"POST /message"`) a následně periodicky ohlašuje svou aktivitu `"POST /pooling"`. Pro předávání strukturovaných dat (obr. 4) využívá formát JSON[5]. Průzkum webového serveru centrály nezjistil žádné jiné vstupní body řídícího centra (pouze `/messages` a `/pooling`).

Současně s HTTP komunikací generuje požadavky na různé záznamy DNS v doméně `sWITH. VexoCIde.orG` (obr. 2). Testováním bylo zjištěno, že v doméně byl vytvořen *doménový koš*:

```
*.sWITH.VexoCIde.orG A 202.254.186.190
```

[5] JavaScript Object Notation – odlehčený formát pro výměnu dat

Time	Source	Destination	Protocol	Info
09:30:59.89	172.16.1.1	195.140.243.4	TCP	59202 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=61368076 TSER=0 WS=7
09:30:59.91	195.140.243.4	172.16.1.1	TCP	http > 59202 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=255254849 TSER=61368076 WS=6
09:30:59.91	172.16.1.1	195.140.243.4	TCP	59202 > http [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=61368093 TSER=255254849
09:30:59.91	172.16.1.1	195.140.243.4	HTTP	POST /message HTTP/1.1
09:30:59.93	195.140.243.4	172.16.1.1	TCP	http > 59202 [ACK] Seq=1 Ack=655 Win=7104 Len=0 TSV=255254854 TSER=61368093
09:31:00.29	195.140.243.4	172.16.1.1	HTTP	HTTP/1.1 200 OK
09:31:00.29	172.16.1.1	195.140.243.4	TCP	59202 > http [ACK] Seq=655 Ack=121 Win=5888 Len=0 TSV=61368473 TSER=255254944
09:31:00.29	172.16.1.1	195.140.243.4	HTTP	POST /message HTTP/1.1
09:31:00.31	195.140.243.4	172.16.1.1	TCP	http > 59202 [ACK] Seq=121 Ack=1292 Win=8448 Len=0 TSV=255254949 TSER=61368474
09:31:00.67	195.140.243.4	172.16.1.1	HTTP	HTTP/1.1 200 OK
09:31:00.67	172.16.1.1	195.140.243.4	HTTP	POST /message HTTP/1.1
09:31:00.69	195.140.243.4	172.16.1.1	TCP	http > 59202 [ACK] Seq=241 Ack=1452 Win=9728 Len=0 TSV=255255044 TSER=61368855
09:31:01.06	195.140.243.4	172.16.1.1	HTTP	HTTP/1.1 200 OK
09:31:01.06	172.16.1.1	195.140.243.4	HTTP	POST /message HTTP/1.1
09:31:01.08	195.140.243.4	172.16.1.1	TCP	http > 59202 [ACK] Seq=361 Ack=1615 Win=11072 Len=0 TSV=255255140 TSER=61369240
09:31:01.18	195.140.243.4	172.16.1.1	HTTP	HTTP/1.1 200 OK
09:31:01.22	172.16.1.1	195.140.243.4	TCP	59202 > http [ACK] Seq=1615 Ack=481 Win=5888 Len=0 TSV=61369404 TSER=255255167
09:31:32.89	172.16.1.1	195.140.243.4	HTTP	POST /polling HTTP/1.1
09:31:32.91	195.140.243.4	172.16.1.1	TCP	http > 59202 [ACK] Seq=481 Ack=1739 Win=11072 Len=0 TSV=255263099 TSER=61401076
09:31:32.91	195.140.243.4	172.16.1.1	HTTP	HTTP/1.1 200 OK
09:31:32.91	172.16.1.1	195.140.243.4	TCP	59202 > http [ACK] Seq=1739 Ack=601 Win=5888 Len=0 TSV=61401093 TSER=255263100
09:31:59.89	172.16.1.1	147.231.25.14	DNS	Standard query A X4DDEO1ef.sWitCh.VEXOcide.orG
09:32:01.94	147.231.25.14	172.16.1.1	DNS	Standard query response A 202.254.186.190
09:32:05.89	172.16.1.1	195.140.243.4	HTTP	POST /polling HTTP/1.1
09:32:05.91	195.140.243.4	172.16.1.1	HTTP	HTTP/1.1 200 OK
09:32:05.91	172.16.1.1	195.140.243.4	TCP	59202 > http [ACK] Seq=1863 Ack=721 Win=5888 Len=0 TSV=61434093 TSER=255271349
09:32:38.89	172.16.1.1	195.140.243.4	HTTP	POST /polling HTTP/1.1
09:32:38.91	195.140.243.4	172.16.1.1	HTTP	HTTP/1.1 200 OK
09:32:38.91	172.16.1.1	195.140.243.4	TCP	59202 > http [ACK] Seq=1987 Ack=841 Win=5888 Len=0 TSV=61467092 TSER=255279600
09:32:59.89	172.16.1.1	147.231.27.173	DNS	Standard query A X4DDEO22B.SwitcH.vEXOcIDe.oRg
09:33:00.12	147.231.27.173	172.16.1.1	DNS	Standard query response A 202.254.186.190
09:33:11.89	172.16.1.1	195.140.243.4	HTTP	POST /polling HTTP/1.1
09:33:11.91	195.140.243.4	172.16.1.1	HTTP	HTTP/1.1 200 OK
09:33:11.91	172.16.1.1	195.140.243.4	TCP	59202 > http [ACK] Seq=2111 Ack=961 Win=5888 Len=0 TSV=61500091 TSER=255287850
09:33:44.90	172.16.1.1	195.140.243.4	HTTP	POST /polling HTTP/1.1
09:33:44.91	195.140.243.4	172.16.1.1	HTTP	HTTP/1.1 200 OK
09:33:44.91	172.16.1.1	195.140.243.4	TCP	59202 > http [ACK] Seq=2235 Ack=1081 Win=5888 Len=0 TSV=61533091 TSER=255296100
09:33:59.89	172.16.1.1	147.231.25.16	DNS	Standard query A X4dDeO267.sWITCH.VexoCIde.orG
09:33:59.93	147.231.25.16	172.16.1.1	DNS	Standard query response A 202.254.186.190
09:34:14.79	172.16.1.1	195.140.243.4	TCP	59202 > http [FIN, ACK] Seq=2235 Ack=1081 Win=5888 Len=0 TSV=61562972 TSER=255296100
09:34:14.81	195.140.243.4	172.16.1.1	TCP	http > 59202 [FIN, ACK] Seq=1081 Ack=2236 Win=11072 Len=0 TSV=255303574 TSER=61562972
09:34:14.81	172.16.1.1	195.140.243.4	TCP	59202 > http [ACK] Seq=2236 Ack=1082 Win=5888 Len=0 TSV=61562989 TSER=255303574

Obr. 2: Výpis záznamu síťové aktivity

S jistou pravděpodobností se mohlo jednat o postranní kanál pro předávání informací mezi řídícím centrem a spuštěným malwarem [25]. Skutečný účel těchto dotazů však objasnila až další analýza.

Výstup z programu (obr. 3) duplikuje informace zachycené pomocí sledování síťové aktivity. Navíc od ní však obsahuje všechna data v originálním znění, ve které je úvod každé zprávy (prvních 32b) před odesláním zašifrován (obr 4). Předpokládáme, že tento výstup a plain-textová část síťové komunikace byly v programu ponechány záměrně pro účely cvičení.

Další akce které malware na stroji po svém spuštění provádí mohou být zjištěny ze záznamu systémových volání (obr. 5).

3.2 Úroveň 2: Co to k čertu...

Dále byly provedeny základní forenzní kroky vedoucí k identifikaci obsahu spustitelných souborů nástroji `file`, `ldd`, `strings`, `objdump` a online službou `virustotal.com`. Vzorky nebyly analyzovány žádným online sandboxem[6].

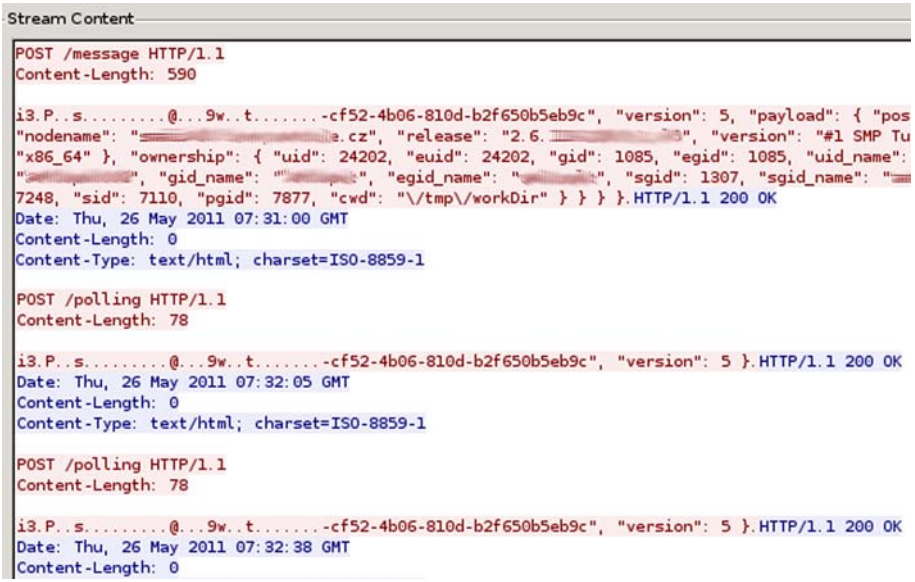
[6] CWSandbox, Anubis Sandbox a Norman Sandbox, ...

```
"running wopr_build_v6_debian32.ANALY_GLASGOW"
The peace bringer exited with: 0
"running wopr_build_centos64.ANALY_GLASGOW"
message_send(): { "pid": 12897, "uuid": "8618d203-c4a1-4393-b743-914c1c7fedcb",
    "version": 5,
 "sitename": "ANALY_XXX", "vo": "xxxxx", "payload": { "posix": { "uname":
    { "sysname": "Linux",
 "nodename": "xxxxxxxxx.xxxx.xxxxxxxx.cz", "release": "2.6.xx-1xx.x2.1.xxx",
    "version": "#1 SMP
 Tue xxx 4 12:47:36 EST 2011", "machine": "x86_64" }, "ownership": { "uid":
    24202, "euid":
 24202, "gid": 1085, "egid": 1085, "uid_name": "xxxxxxxx002", "euid_name":
    "xxxxxxxx002",
 "gid_name": "xxxxxxxt", "egid_name": "xxxxxxxt", "sgid": 1307, "sgid_name":
    "xxxxx" },
 "process": { "pid": 12897, "ppid": 12891, "sid": 12569, "pgid": 12569, "cwd":
    "\/scratch\/83947
 56.xxxxxx.xxxx.xxxxxxxx.cz\/cxxxxxg_Brn12797\/xxxxx3\/
    Pxxxx_Pxxxx_12826_1306315250\/Pxxxxxxx_12
 41710574_1306315252\/workDir" } } } }
Encrypting...
message_callback
Decrypting...

****************************** Got it ************************

polling_message_send: { "pid": 12897, "uuid":
    "8618d203-c4a1-4393-b743-914c1c7fedcb",
    "version": 5,
 "sitename": "ANALY_XXX", "vo": "xxxxx" }
Encrypting...
Polling callback
Decrypting...
polling_callback: got response object without an input buffer
resolve_dispatch
resolve_cb
polling_message_send: { "pid": 12897, "uuid":
    "8618d203-c4a1-4393-b743-914c1c7fedcb", "version": 5,
 "sitename": "ANALY_XXX", "vo": "xxxxx" }
Encrypting...
Polling callback
Decrypting...
polling_callback: got response object without an input buffer
```

Obr. 3: Ukázka `tmp.stdout`



Obr. 4: Ukázka komunikace s řídícím centrem

```
open("/tmp/some.random.file.move.along", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 13
open("/var/tmp/some.random.file.move.along", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 14
open("/opt/glite/var/tmp/some.random.file.move.along",
  O_WRONLY|O_CREAT|O_TRUNC, 0666) = -1 ENOENT (No such file or directory)
open("/opt/glite/var/log/some.random.file.move.along",
  O_WRONLY|O_CREAT|O_TRUNC, 0666) = -1 ENOENT (No such ...
```

Obr. 5: Ukázka `strace.out`

3.2.1 man file

Oba spustitelné soubory jsou staticky linkované a nejsou z nich odstraněny ladící informace (not stripped).

Statické linkování (static linking) se používá z důvodů snadné přenositelnosti. Takto sestavený program je (až na malé výjimky) nezávislý na dostupnosti potřebných verzí knihoven na cílovém počítači, protože všechen potřebný kód pro vykonání programu včetně všech knihoven je součástí spustitelného souboru. Vytvořený program je sice větší (na disku i v paměti), ale spolehlivější přenositelnost je pro malware výhodou.

Ladící informace (debugging symbols) ve spustitelném programu (jména a relativní adresy proměnných, podprogramů a jejich parametrů, ...) se využijí v případě, že program při svém běhu havaruje (post mortem analýza z core-

```
wopr_build_centos64.ANALY_GLASGOW:
 ELF 64-bit LSB exec, x86-64, statically linked, for GNU/Linux 2.6.9,
    not stripped
wopr_build_v6_debian32.ANALY_GLASGOW:
 ELF 32-bit LSB exec, Intel 80386, statically linked,
    for GNU/Linux 2.4.1, not stripped
```

Obr. 6: Výstup programu `file`

Obr. 7: Výběr tisknutelných sekvencí a ostatní ukázky

dumpu), nebo pokud je nutné jej debugovat za běhu. Pro samotnou činnost programu však nemají žádný význam a bývají odstraňovány (`man strip`) v koncové fázi vývoje software (release build). V případě malware umožní nebo ztíží jeho případnou analýzu.

3.2.2 man strings

Rychlou nápovědou o funkcích nebo činnosti viru mohou být sekvence tisknutelných znaků které binární soubor obsahuje. Na rozdíl od reálných virů, poskytují již první nalezené čitelné řádky[7] wopru znatelnou úlevu (obr. 7).

Ve vzorku byly nalezeny IP adresy veřejně dostupných DNS resolverů následované informacemi o adrese C&C a jeho URL handlerech, formátovací řetězec pro DNS dotazy, pravděpodobně jméno nějakého souboru, textové části shell skriptů testujících možnost zápisu do adresáře, seznam adresářů a skripty pro

[7] čti „Přicházíme v míru, google.com“;)

```
1338: 08052ce0   227 FUNC    GLOBAL DEFAULT    3 evbuffer_encrypt
1536: 0805e700   570 FUNC    GLOBAL DEFAULT    3 evbuffer_add
1695: 0804ce50  4871 FUNC    GLOBAL DEFAULT    3 aes_decrypt
2280: 0806dd40   360 FUNC    GLOBAL DEFAULT    3 evhttp_make_request
2521: 08069430    11 FUNC    GLOBAL DEFAULT    3 evhttp_request_get_respon
```

Obr. 8: Ukázka tabulky symbolů `egi.ssc5.wopr`

> *libevent is an event notification library for developing scalable network servers. The libevent API provides a mechanism to execute a callback function when a specific event occurs on a file descriptor or after a timeout has been reached. Furthermore, libevent also support callbacks due to signals or regular timeouts.*

Obr. 9: Specifikace projektu libevent

zavedení úlohy do lokálních plánovačů `at` a `cron`. Práci s textovými daty mohou ulehčit například nástroje typu `bashitsu` [34].

Takto „náhodně nalezené“ informace mohou pomoci nalézt ostatní infikované stroje, vylepšit stávající pravidla IDS a IPS systémů a zvolit další postup při analýze bezpečnostního incidentu a navrhovaní dalšího postupu jeho řešení.

3.2.3 man objdump

Pomocí programů `objdump`, `readelf`, ..., je dále možné získat (kromě množství náhodných hexadecimálních čísel) přehled o rozložení informací ve spustitelném souboru, jména a adresy používaných proměnných a podprogramů, a to jak interních, exportovaných nebo importovaných [26]. Tyto informace nám dále mohou zlepšit představy o účelu a povaze malware.

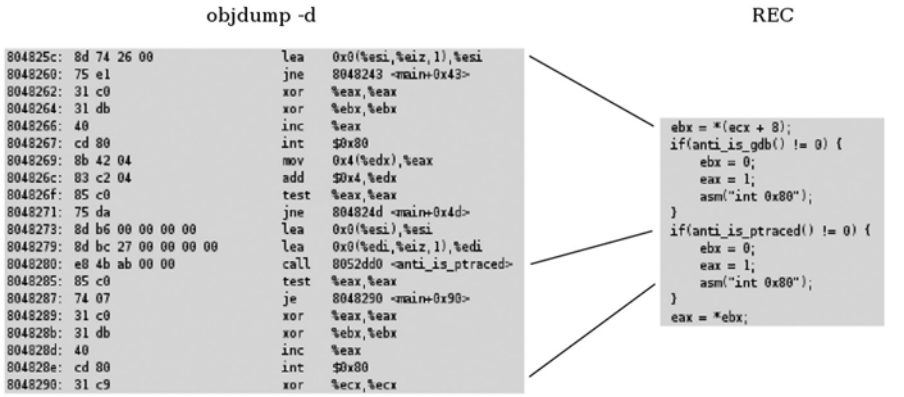
Z obrázku 8 jsou vidět některé používané funkce. Pomocí internetových vyhledávačů je pak možné zjistit, že funkce pocházejí pravděpodobně z knihovny `libevent`[8] z dílny bezpečnostního specialisty Nielse Provose. Podle popisu projektu (obr. 9) se tato knihovna výborně hodí pro tvorbu různých programů malware nevyjímaje.

Zajímavé jsou například funkce `aes_decrypt` a `evbuffer_encrypt`, dokládající pravděpodobnou metodu šifrovaní datového toku pomocí šifry AES. V dokumentaci API knihovny `libevent` tato funkce `evbuffer_encrypt` není.

3.2.4 man imagination

Při zkoumání neznámých souborů (např. nespustitelných) můžeme využít i různých druhů vizualizace [27]. V některých případech je potřeba obejít některou

[8] http://monkey.org/ provos/libevent/



Obr. 10: `objdump` vs `rec`

z forem použité obfuskace. `wopr` ve verzi 5.0 neobsahuje žádnou z nich:

- falešné magic hlavičky a přípony, neviditelná adresářová struktura,
- odstranění nebo přidání bílých znaků, různá kódování (base64, url, xor, concat ...),
- zkomprimování skutečného obsahu algoritmem gzip nebo specializovanými packery,
- nebo zašifrování částí malware[9].

3.3 Úroveň 3: Mezi řádky

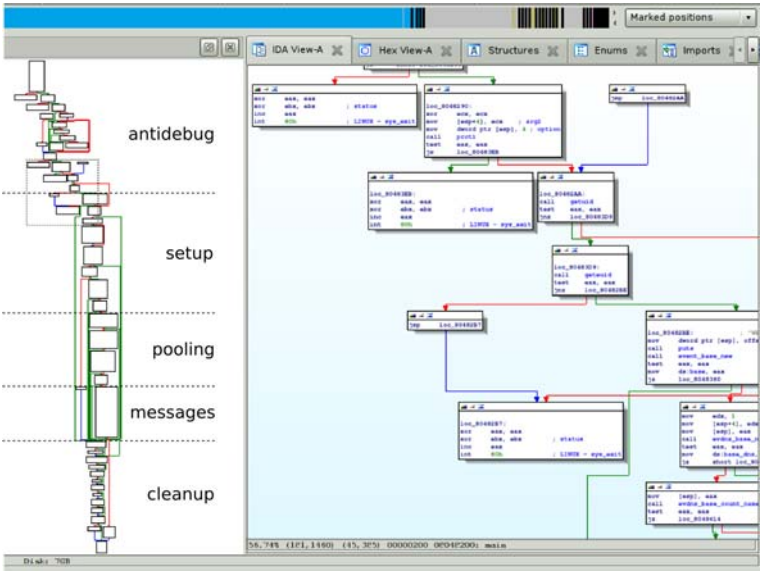
Další úrovní analýzy zachyceného vzorku bývá disassemblování a zjištění codeflow virového vzorku. Tato úroveň vyžaduje znalosti programování v C [24] a Assembleru [3, 26], překladačů a operačních systémů na Von Neumannovských architekturách [19, 20] a postupů používaných při konstrukcích malware [8, 9, 16, 18, 30].

Pro získání člověkem čitelného strojového kódu[10] může posloužit program `objdump` z kapitoly 3.2.3 nebo některý ze specializovanějších programů určených pro reverzní inženýrství [4, 11, 14].

V obou případech (obr. 10) je výsledkem disassemblingu textový soubor obsahující strojový kód programu. Práce s tímto souborem je poměrně náročná a v případě větších projektů prakticky nemožná i přes to, že `rec` [4] generuje kód vyšší úrovně (rozpoznané řídící konstrukce programu jsou vyjádřeny v jazyce C).

[9] `wishmaster`

[10] zjevný protimluv;)



Obr. 11: Ukázka grafického rozhraní IDA Pro 6.0 demo

Alternativou může být použití nástrojů poskytujících interaktivní vizualizaci typu OllyDbg[11], ty poskytují k získanému kódu grafické rozhraní s možností vyhledávání a pohodlného sledování skoků při větvení programu. V každém případě se stále jedná o práci s plochou strukturou programu.

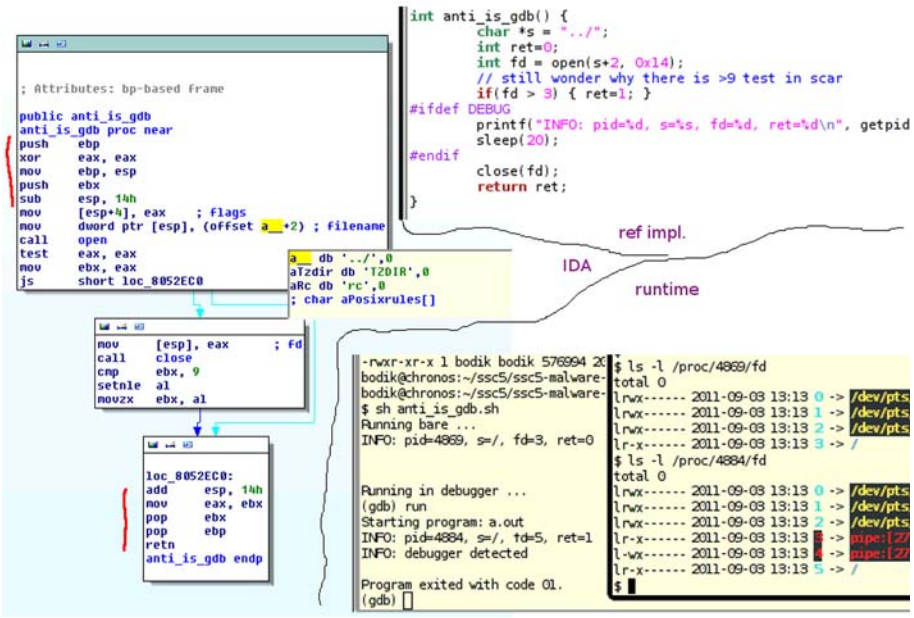
Naštěstí v dnešní době sofistikovaného malware [17, 16] existují i specializované nástroje, které jednak poskytují vynikající analytické funkce pro získání dat z předložených programů a navíc s nimi umožní pracovat ve skutečně interaktivním grafickém rozhraní. Jedním z nich je IDA Pro [1] – pokročilý disassembler, jehož hlavní předností je interaktivní grafické zobrazení bloků strojového kódu zkoumaného programu provázané v místech větvení toku programu a provázání veškerých dostupných informací získaných z jednotlivých segmentů programu (text, data, got, plt, bss, ...).

Tímto nástrojem jsme dokázali identifikovat části malware `egi.ssc5.wopr` a určit jeho možnosti.

3.3.1 Antidebug

Úvodní část `egi.ssc5.wopr` obsahuje několik antidebugovacích podprogramů. Tyto techniky jsou určené pro znemožnění jeho spuštění v řízeném prostředí

[11] pomineme-li fakt, že OllyDbg je hlavně debugger



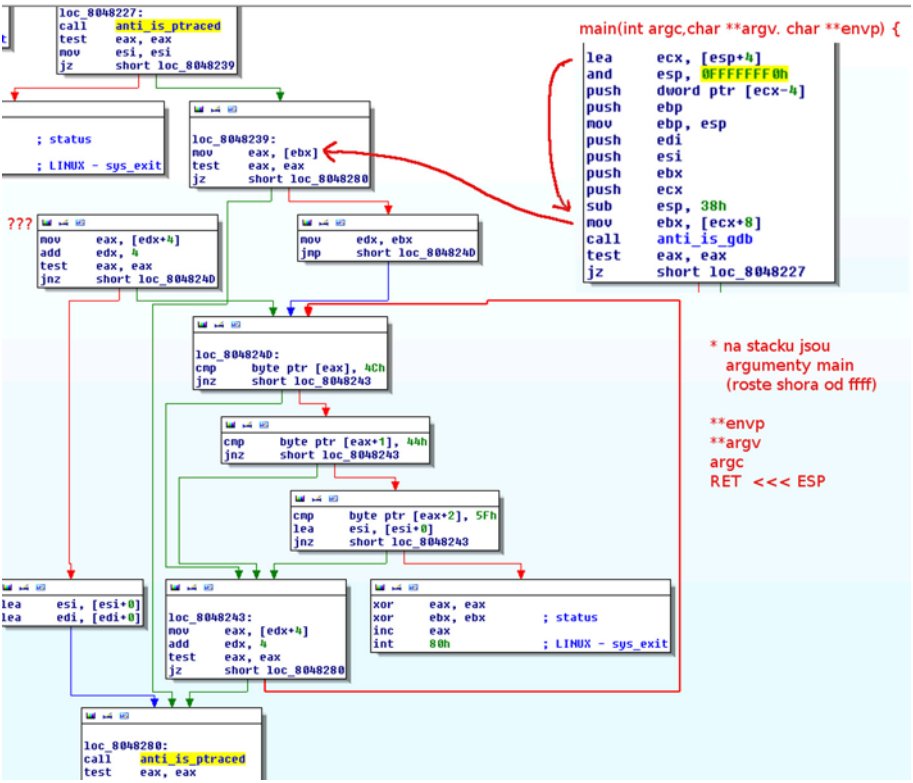
Obr. 12: Analýza `anti_is_gdb()`

a ztížit jeho případnou analýzu. Mezi další techniky tohoto typu patří obfuskace, packing, falešné větve programu a přebytečné skoky, detekce běhu uvnitř debugeru (`IsDebugerPresent`, `OutputDebugString`, `FindWindow`, ...), prohledávání registrů, generování výjimek, kontrola velikosti různých souborů na disku, apod. V této oblasti se vynalézavosti meze nekladou.

Podprogram `anti_is_gdb()` testuje číslo prvního volného filedeskriptoru. Volání rutiny je umístěno hned na začátku programu a předpokládá se, že v té chvíli budou otevřeny pouze 3 základní I/O proudy (0 – stdin, 1 – stdout, 2 – stderr) a nově otevřený FD by tedy neměl být vyšší než 3. Pokud tomu tak je, naznačuje to, že byl program spuštěn v nestandardním prostředí a je pravděpodobně debugován. Např. při spuštění v debugeru GDB, zdědí laděný proces otevřené STD proudy od svého rodiče a navíc má roury (IPC pipes) pro komunikaci s ním. Prvním volným FD je tedy 5 (při spuštění v DDD je první volný FD 7).

V `egi.ssc5.wopr` (obr. 12) je testován první volný deskriptor podmínkou `fd>9` (`cmp %ebx, 9`), což může souviset se specifiky cílového prostředí gridu (job piloting) nebo chybu při implementaci ochrany[12] vývojem v některém z pokročilých IDE (Netbeans, ...).

[12] very unlikely...



Obr. 13: Analýza anti-test for ENV

Podprogram `anti_is_ptraced()` testuje přítomnost aktivního debugeru využitím faktu, že každý proces nemůže být současně trasován více než jednou. Malware vytvoří potomka, který se jej pokusí trasovat a návratovou hodnotou svému rodiči předá informaci o výsledku testu. Při analýze podprogramu byla sestavena referenční implementace.

Sekce anti-test for ENV (obr. 13) má za úkol zkontrolovat proměnné prostředí a vyhledat v něm proměnnou začínající znaky `LD_`. Tato technika zřejmě slouží k odhalení běhu v některém z linuxových sandboxů založených na `LD_PRELOAD`.

I přesto, že zjistit účel této oblasti kódu je poměrně snadný (porovnávání řetězce se znaky `0x4C 0x44 0x5F` v cyklu je z grafu dobře vidět), je tato ukázka vhodná pro osvěžení znalostí instrukční sady x86 a způsobu uložení vícerozměrných polí v paměti.

Zásobník (paměť sloužící pro uložení řídících dat programu) roste od vyšších adres k nižším. Při volání funkce se na něj nejdříve uloží argumenty volané funkce. Ukládají se od konce. Druhým a třetím argumentem fce `main` jsou pole řetězců, ve skutečnosti jsou to pointery, které ukazující na pole pointerů, které ukazují na skutečný obsah [24].

Při práci s těmito strukturami se používají instrukce `mov dst,[src]`, přičemž `[]` znamenají dereferenci adresy `src` – tedy do `dst` je nahrán obsah paměti z adresy `src`. Naproti tomu instrukce `lea dst,[src]` nahraje do `dst` adresu (ne obsah paměti), kterou představuje `src`. Pro tuto instrukci neznamenají `[]` dereferenci adresy. Instrukce `mov` je určena pro manipulaci s daty na adrese, zatímco `lea` je určena pro vypočítávání adres. V obou případech se mohou uvnitř `[]` objevit aritmetické výrazy.

Kód související s tímto antidebug blokem začíná hned v první instrukci programu. Zde instrukce `lea ecx,[esp+4]`[13] vypočítá adresu, na které leží první argument funkce `main` (parametr `argv`). Druhá *související* instrukce leží až za prologem funkce `main`. `mov ebx,[ecx+8]` nahraje obsah z adresy `ecx+8`[14] do registru `ebx`, ten tedy obsahuje adresu prvního pointeru v poli řetězců, které představuje proměnné prostředí spuštěného programu.

Samotný blok začíná až dále v programu za voláním funkce `anti_is_ptraced`. Jedná se o jednoduchou smyčku provádějící porovnání prvních 3 znaků z řetězce proměnné prostředí, která v případě nalezení shody celý program ukončí. V poli pointerů na jednotlivé řetězce se nakonec iteruje pomocí registru `edx` a blok končí opakovaným voláním funkce `anti_is_ptraced`.

Zbylý antidebug kód zakáže pomocí volání `prctl(PR_SET_DUMPABLE,0)` vygenerování obsahu procesu v případě jeho pádu ve snaze zanechávat po sobě co nejméně stop a zkontroluje UID pod kterým je `egi.ssc5.wopr` spuštěn. Pokud je to uživatel s UID=0 je proces ukončen.

3.3.2 Nastavení

Po ochranách proti analýze začne vytvářet objekty pro další běh za pomoci knihovny `libevent`.

1. Nastaví knihovně seznam DNS serverů z konfigurace systému, případně použije IP adresy veřejných DNS serverů společnosti Google.
2. Naplánuje jednorázovou událost, která malware sama ukončí po uplynutí 14 dnů.

[13] na stacku tedy přeskočí návratovou adresu (4b).

[14] na stacku tedy přeskočí počet argumentů `main` (4B) + pointer na pole řetězců s parametry programu (4B).

3. Založí opakovanou událost, která periodicky ($t = 60$ s) provede unikátně vygenerovaný DNS dotaz do domény `sWITH.VexoCIde.orG`. Dotazy obsahují jméno které je odvozeno z aktuálního času (kvůli cache) a odpověď je porovnána s konstantní hodnotou. Je zajímavé, že tato hodnota je v *network byte order* reprezentována jako `0xcafebabe`. Domníváme se, že jde o další možný antidebug kód, případně *deadman switch*. Pokud by se v průběhu cvičení vyskytly nějaké nečekané problémy tak by vhodná změna na NS serveru dané zóny dokázala celý projekt do několika vteřin řízeně ukončit.

3.3.3 Příjem příkazů

V další fázi své inicializace zaregistruje wopr opět periodickou akci. V tomto případě se jedná o pravidelné kontakty na centrálu v intervalu 0x21 sekund. Zprávy `"POST /pooling"` (obr. 4) obsahují vždy pouze standardní hlavičku {`pid:, uuid:, version:,` ...} (viz `message_new`). Od serveru je v callbacku události (fce `pooling_callback`) očekávána odpověď ve formátu JSON a po jejím obdržení a úspěšném rozparsování je zpráva předána do podprogramu `command_dispatcher`.

Před odesláním a po přijmu každé zprávy, je úvodní část (první dva 16B bloky zprávy) zašifrována algoritmem AES256 v módu EBC[15] – `evbuffer_encrypt`. Ačkoli je jméno podprogramu v souladu s názvoslovím knihovny `libevent`, není součástí oficiálního projektu. Tomu nasvědčuje i jeho umístění ve spustitelném souboru (není ve stejné oblasti jako ostatní části knihovny). Jako klíč je použit kontrolní součet `sha256` řetězce `omgwtfbbqidkfaiddqd`. Funkcí *Save as C array* programu Wireshark, byla tato skutečnost ověřena referenčním programem, který je schopen dešifrovat zachycenou komunikaci mezi malware a C&C. Domníváme se, že pro šifrování byla použita implementace od Christophera Devina.

Heslo bylo identifikováno jako složenina několika IT zkratek a je vidět již ve výstupu `strings` z kapitoly 3.2.2:

- `omg` – Oh my god!
- `wtf` – What the fuck?
- `bbq` – Barbecue (zvolání, když obyčejné wtf nestačí)
- `idkfa` – Doom cheat – všechny zbraně a klíče
- `iddqd` – Doom cheat – nesmrtelnost

Podprogram `command_dispatcher` se pokusí rozparsovat odpověď od centrály a jsou v něm implementovány reakce na příkazy:

[15] Každý blok je zašifrován stejným klíčem, viz EBC vs. CBC.

- `network_info` – viz kap. 3.3.4,
- `system_info` – viz kap. 3.3.4,
- `halt`, `kill` – oba ukončí provádění programu pomocí funkce `clean_kill`, který uvolní objekty knihovny `libevent` a poté zavolá `exit(0)`,
- `shell` – obdržený příkaz spustí pomocí volání `system`.

3.3.4 Úvodní přihlášení do botnetu

Posledním krokem před spuštěním knihovny `libevent` je shromáždění základních informací o systém kde malware běží a jejich odeslání v požadavcích `"POST /messages"`. Informace o systému sbírají níže vyjmenované podprogramy a jejich výsledek je po skončení vždy odeslán jednotlivě.

- `payload_networked_devices_to_json` – zjistí hostname a sestaví seznam síťových zařízení (`getifaddrs`) včetně podrobných informací,
- `payload_postix_info_to_json` – zjistí informace o identitě aktuálního procesu: `uname`, `getuid`, `geteuid`, `getgid`, `getegid`, `getpwuid`, `getgrgid`, `getgroups`, `getpid`, `getppid`, `getsid`, `getgpid`, `getcwd`,
- `payload_get_info_environment` – zjistí informace o proměnných prostředí `VO`, `Site`, `ROC`, `X509_USER_PROXY`
- `payload_look_for_writeable_dirs` – testem vyzkouší a nahlásí možnosti zápisu ve vyjmenovaných adresářích (kap. 3.2.2)
- `payload_test_cron_at` – název neodpovídá implementaci. Funkce vytiskne na obrazovku řetězec `"** GOT IT **"` a potom opět zavolá `payload_look_for_writeable_dirs`. Domníváme se, že toto je *nedokončená* větev programu, v malware sice jsou implementovány funkce `payload_test_try_cron` a `payload_test_try_at`, ale ty nejsou nikde volány.

Po dokončení této fáze je řízení programu předáno knihovně `libevent`[16].

3.3.5 `egi.ssc5.wopr.32` vs. `egi.ssc5.wopr.64`

Porovnáním 32b a 64b verze byly zjištěny drobné rozdíly. 64b verze neobsahuje úvodní antidebug ochrany, navíc obsahuje různé ladící výpisy. `command_dispatcher` má drobně odlišnou posloupnost volání a obsahuje rozpoznání dalších příkazů (`credential_info` a `cron_at_info`), obě větve však vedou pouze na blok `printf("Not implemented yet")`. Implementace funkcí `payload_test_try_cron` a `payload_test_try_at` pouze vrací `-1`, ve 32b verzi jsou funkce implementovány celé, ale jsou nepoužívané.

[16] `event_base_dispatch`

3.4 Úroveň 4: Drakova hlava

Pokud by se jednalo o skutečný virus, následovaly by hlubší pokusy o infiltraci analyzovaného botnetu ať už dlouhodobým během viru v řízeném prostředí a sledováním jeho činnosti nebo přímými pokusy vyřazení botnetu z provozu přímým útokem na C&C či využitím deadman switche ve spolupráci s registrátorem domény.

Pokud by se nepodařilo odhalit šifrovací klíč, mohly by být informace z kapitoly 3.1 použity ke kryptoanalýze typu known-plaintext (zřejmě za využití prostředků výpočetního prostředí EGI :).

Výsledky rozboru vzorků by umožnily dále zkoumat dopady běhu viru na napadených strojích, nabízí se prohlídka process accountingu za účelem zjištění příkazů spuštěných přes `command_dispatcher` nebo hlubší forenzní analýza napadených uzlů.

Vzhledem k tomu, že se jednalo o cvičení, nebyly podniknuty žádné tyto kroky.

4 A co na to Jan Tleskač™?

I přes to, že analýza malware nebyla hlavním úkolem cvičení, domníváme se, že i s touto části autoři počítali a sestavený vzorek byl velmi pečlivě připraven tak, aby poskytl vzdělávací informace na všech úrovních analýzy.

Žádný z antivirů ve službě http://virustotal.com neidentifikoval možnost hrozby i přes přítomnost neobfuskovaných antidebugovacích podprogramů.

Formát JSON je velmi výhodným formátem pro předávání strukturovaných dat typu asociativní pole (hash) a k jejich ukládání se dají využívat NoSQL databáze (např. Redis [31]).

Při práci mohou být s výhodou používány různé taháky [32] nebo nástroje pro filtrování[34] a management získaných informací [33]. Je dobré znalosti průběžně konzultovat se světovými vyhledávači. Proces analýzy mohou značně usnadnit profesionální komerční nástroje nebo alespoň jejich nekomerční verze.

Domníváme se, že malware by mohl být schopný působit i v prostředí s protokolem IPv6.

5 Závěr

Při řešení tohoto cvičení postupoval řešící tým podle obecných pravidel pro šetření bezpečnostních incidentů i specifických pravidel EGI. V úvodní fázi bylo nutné izolovat napadené uzly a jejich forenzní analýzou získat základní informace o vzniklém incidentu, nejlépe včetně vzorku virové úlohy. Podle nalezených dat

dále identifikovat ostatní závadné úlohy ve výpočetním prostředí, nalézt uživatele který do systému úlohu vložil a zablokovat mu přístupové údaje.

Vzhledem k provázanosti jednotlivých výpočetních center bylo nutné sdílet informace o provedených akcích podle předepsaných postupů s ostatními členy EGI a týmem EGI CSIRT, tento proces byl hlavním úkolem cvičení. Postupně se podařilo shromáždit balík dat obsahující 2 virové vzorky včetně jejich spouštěcích skriptů a metadat o daných úlohách. Balík byl poté podroben analýze reverzním inženýrstvím, která měla za úkol rekonstrukci útoku a zjištění jeho dopadu na výpočetní prostředí.

Pro analýzu byly využity aktivní i pasivní metody. Aktivní část zahrnovala spuštění úlohy v řízeném prostředí interaktivního honeypotu za účelem získání záznamu o síťové aktivitě úlohy a její interakci s operačním systémem. Pasivní část zahrnovala dekompilaci nalezených virových vzorků a z nich zjištění codeflow viru.

Provedené šetření zjistilo pravděpodobné chování viru, identifikovalo použité knihovny, programovací styl a pomohlo získat znalosti, které by vedly k infiltraci do postaveného botnetu s pasivní možností sledování aktivit útočníka. Dále vedlo k navržení dalšího možného postupu forenzní analýzy napadených uzlů gridu.

Při analýze byl použit sw IDA Pro, jehož hlavní předností je interaktivní grafické zobrazení bloků strojového kódu zkoumaného programu provázané v místech větvení toku programu. (if, switch, call, goto, ...). Vzhledem k tomu, že zachycený virus byl uměle vytvořený pro potřeby bezpečnostního cvičení SSC5, jeho kód nebyl nijak obfuskován a programovací styl byl na vysoké úrovni, podařilo se relativně rychle a snadno identifikovat používané podprogramy.

Zachycený malware je *event driven network server* napsaný pomocí knihovny `libevent` a jako C&C používá HTTP echo based kanál. V něm jsou implementovány základní kameny pro ochranu komunikace (symetrické šifrování statickým klíčem). Celkově se jedná o jednoduchý *exec pad* jehož hlavním úkolem je spojení s řídícím serverem, sběr informací o napadeném prostředí a vykonávání příkazů botmastera pomocí systémového volání `system`. Malware byl vyroben v několika buildech, z nichž každý obsahoval drobné odchylky v dostupné funkcionalitě. 32bitová mutace obsahovala i antidebug prvky, které měly znesnadnit runtime analýzu. Implementované bezpečnostní prvky byly upraveny tak, aby usnadnily analýzu, ale zachovaly výukovou hodnotu. Autoři zřejmě počítali i s hloubkovou analýzou softwaru a umístili na několik míst vtipné poznámky tak, aby analýza byla i zábavná.

Literatura a odkazy

[1] *IDA Pro: multi-processor disassembler and debugger*
http://www.hex-rays.com/idapro/

[2] ISC Cheat sheet http://isc.sans.edu/presentations/iscflyer.pdf

[3] Iczelion's Win32 Assembly Tutorials
http://win32assembly.online.fr/tut1.html

[4] REC Studio 4 – Reverse Engineering Compiler
http://www.backerstreet.com/rec/rec.htm

[5] k0fein: *EGEE – Enabling Grids for E-science-E*

[6] Josef Kadlec: *Forenzní analýza Unixových systémů*
Diplomová práce, http://fim.uhk.cz, 2006.

[7] Radoslav Bodó: Kovářova kobyla. . . : případová studie forenzní analýzy OS Linux, *Sborník konference Europen.cz*, jaro 2007, Jablonné v Podještědí ISBN 978-80-86583-12-9.

[8] MazeGen: *Obfuskace strojového x86 kódu.*

[9] Bojan Zdrnja: *ISC Diary: Analyzing an obfuscated ANI exploit.*
http://isc.sans.org/diary.html?storyid=2826

[10] Yshey Eset.sk: *10 things to learn from malware.*

[11] http://www.joineset.cz/w4-download/List_kandidatovi_cz.pdf

[12] Sam Stover, Matt Dickerson: *Using memory dumps in digital forensics*; login:, vol. 30, no. 6, pp. 43–48.

[13] Jan Goebel, Jens Hektor, Thorsten Holz: *Advanced honeypot-based intrusion detection*; login: v. 31 n. 6.

[14] Chaos Golubitsky: *Simple software flow analysis using GNU CFlow.* ;login: v. 31 n. 2.

[15] Craig A. Schiller, Jim Binkley, David Harlet, Gadi Evron, Tony Bradley, Carsten Willems, Michael Cross: *Botnets: The killer web app.* ISBN 978-1-59749-135-8.

[16] Dave Ditrich, Sven Dietrich: *Command and control structures in malware: from handler/agent to P2P*; login: vol. 32 no. 6.

[17] bodik and c++: *InSecurity 2009.* EurOpen.cz, Štědronín 2009.

[18] Alexander Sotirov, Mark Dowd: Bypassing Browser Memory Protections: Setting back browser security by 10 years. *Proceedings of BlackHat Conference 2008.*

[19] Shawn Moyer: (un)Smashing the stack: Overflows, Countermeasures and the Real World, *Proceedings of BlackHat Conference 2008.*

[20] Radoslav Bodó: Jak se smaží zásobník: Esej na téma buffer overflow, včera dnes a zítra, *Sborník konference Europen.cz*, jaro 2009, Praděd. ISBN 978-80-86583-16-7.

[21] Dave Dittrich: Basic Steps in Forensic Analysis of Unix Systems. http://staff.washington.edu/dittrich/misc/forensics/

[22] Ivor Kollár: *Forensic RAM dump image analyser.* Diplomová práce, http://mff.cuni.cz, 2010.

[23] xorl: *More GDB Anti-Debugging.* http://xorl.wordpress.com/2009/01/05/more-gdb-anti-debugging/

[24] Pavel Herout: *Učebnice jazyka C.* ISBN 978-8-7232-383-8.

[25] Erik Couture: *Covert Channels.* http://www.sans.org/reading_room/whitepapers/detection/covert-channels_33413

[26] M. Dobšíček, R. Ballner: *Linux bezpečnost a exploity.* ISBN 80-7232-243-5.

[27] Conti: *Security data visualization.* ISBN 978-1-59327-143-5.

[28] http://www.swansontec.com/sregisters.html

[29] http://www.skullsecurity.org/wiki/index.php/Simple_Instructions

[30] Chris Wysopal: Detecting „Certified Pre-owned“ Software and Devices, *Conference BlackHat2009.*

[31] Karel Minařík: *Redis: The AK-47 of Post-relational Databases.* http://europen.cz/Proceedings/38/Redis.The.AK-47.of.Post-relational.Databases.KarelMinarik.2011.pdf

[32] https://home.zcu.cz/ bodik/cheatsheets/

[33] http://freemind.sourceforge.net

[34] http://code.google.com/p/bashitsu/

Bezpečnosť v kontexte RIA technológii

Juraj Michálek

E-mail: juraj.michalek@sinusgear.com

Odkiaľ prišla skratka RIA

Rich Internet Application (RIA) sa stalo označením pre web aplikácie, ktoré prekračovali rámec bežného statického webu. Toto označenie sa začalo objavovať začiatkom roku 2007 v súvislosti s technológiami Adobe Flash, Microsoft Silverlight a Sun Java FX.

Primárnym cieľom týchto technológii bolo posunúť hranice možností webu za schopnosti bežného web prehliadača. Je nutné pripomenúť, že v tej dobe sa stále do veľkej miery vyžadovala podpora pre Internet Explorer 6.

Riešením ako posunúť možnosti prehliadačov bola implementácia pluginu, ktorý by umožňňoval beh aplikácii postavených na určitej platforme. Adobe si zvolilo Flash Platformu, Microsoft vytvoril Silverlight a Sun začal odvodzovať od Javy svoj Java FX.

Najsilnejším hráčom sa tu ukázal Flash Player spolu s open source frameworkom Flex. Flex umožňoval ako prvý na webe vytvárať a zobrazovať veľmi kvalitné grafy. Adobe investovalo zdroje do otvorenia Flex platformy a vytvorilo portál http://opensource.adobe.com. Framework bol uvolnený pod Mozilla public licence. Na jeho začiatku bol uzavretý a postupne boli otvárané jeho časti, a nakoniec bol framework otvorený kompletne. Výhodou tejto platformu bola podpora pre Windows, Mac, Linux.

Microsoft Silverlight nezaznamenal také výrazné rozšírenie. Bariérou bola hlavne nie až tak vydarená podpora pre iné prehliadače než Internet Explorer. Linux bol kompletne vynechaný. Novell síce implementoval Silverlight, ale jednalo sa o nie úplne kompatibilnú verziu so Silverlightom od Microsoftu.

Java FX sa rozvíjala veľmi pomaly a jednalo sa skôr o deklaráciu toho, že aj Sun chcel byť v hre o RIA platformu, ale úspechu sa nedočkal. V poslednej dobe Oracle obnovilo aktivity na Java FX, avšak čo sa týka efektivity a svojich možností, tak nie je toto prostredie tak prepracované ako jeho konkurencia.

S príchodom RIA platformiem prišlo aj niekoľko obmedzení súvisiacich s bezpečnosťou. V tomto texte sa zameriam na aspekty Adobe Flash Platform.

Bezpečnosť v kontexte RIA

Permanentné Cookie

Dlho vyčítaným nedostatkom Flash Playeru bola možnosť tvorby tzv. Persitent Cookie. Flash Player umožňoval vytvárať vlastné cookie, ktoré boli mimo správu web prehliadača. S verziou 10.3 bol tento problém odstránený a používatelia už môžu kontrolovať aj cookie ukladané vrámci Flash Playeru.

Cross-domain

Jednou z problematických otázok pri vývojí webu je tzv. cross-domain. Predstavme si, že web aplikácia sa načíta z web adresy http://www.fi.muni.cz. Táto aplikcia má načítavať dáta o aktualitách a zobrazovať ich. Pokiaľ sa požiadavky na server nachádzajú na tej istej doméne, tak nie je problém. Aplikácia pošle požiadavky a server pošle odpoveť.

V prípade, že by si niekto aplikáciu skopíroval na svoj web a chcel by využívať služby z pôvodnej domény, tak ho obmedzenie Flash Playeru zastavia a nepovolia mu odoslať požiadavku.

Pokiaľ chceme povoliť konzumovanie obsahu z iných domén, je nutné vytvoriť v roote webu súbor crossdomain.xml. V tomto súbore sa vymenujú povolené domény, odkiaľ je možné načítavať obsah.

Taktiež môžeme povoliť konzumáciu dát napríklad pomocou protokolu SOAP.

Príklad súboru crossdomain.xml, ktorý povoľuje prístup z ľubovolnej domény a protokolu SOAP:

```
<cross-domain-policy>
      <allow-access-from˜domain="*"/>
      <allow-http-request-headers-from domain="*" headers="SOAPAction"/>
</cross-domain-policy>
```

Ďalšie detaily je možné nájsť v dokumente Cross-domain policy file specification – http://www.adobe.com/devnet/articles/crossdomain_policy_file_spec.html

REST protokol POST, GET

Pomerne veľkým obmedzením Flash Playeru je jeho schopnosť používať REST protokol. Flash Player je obmedzený web prehliadačom, v ktorom beží. Flash Player odosiela totiž požiadavky cez prehliadač. S bežnou požiadavkou typu GET nie je problém. POST nebol v niektorých starších prehliadačoch podporovaný. Každopádne môžeme tvrdiť, že Flash Player umožňuje odosielanie POST a GET požiadavok.

REST protokoly ale využívajú ďalšie typy ako je PUT, DELETE atp. Tieto sú obvykle používané v službách ako je Amazon S3. Pokiaľ komunikačný protokol so serverom vyžaduje iné požiadavky ako GET a POST, tak ho nie je možné použiť z Flash Playeru. Amazon z tohoto dôvodu implementuje k väčšine požiadaviek aj GET/POST variantu.

Modifikácia hlavičiek HTTP požiadavku

Flash Player umožňuje doplnenie vlastných hlavičiek do HTTP requestu pomocou triedy URLStream. Toto je však limitované a nie vo všetkých prehliadačoch musí fungovať spoľahlivo. Verziu od verzie prehliadača sa podpora môže líšiť.

Načítanie hlavičiek HTTP požiadavku

Silným kameňom úrazu pre používaní REST a HTTP-based protokolov je nemožnosť načítania HTTP hlavičiek, ktoré server v odpovedi odošle. Toto nie je technický problém Flash Playeru, ale prehliadača, v ktorom Flash Player beží. Každopádne v prípade napojenia na cloud služby ako je Amazon, predstavuje toto obmedzenie výrazný problém pri implementácii.

Flash Player sockets

Od Flash Playeru 9.0.124 je možné používať Flash Player aj na priame pripojenie na určitý socket. Toto samozrejme predstavuje pomerne veľké bezpečnostné riziko, pokiaľ by bolo moné použiť pripojenie na socket bez obmedzenia.

Na druhú stranu možnosť využívať sockety prináša veľké možnosti využitia v rôznych konferenčných aplikáciách, či uploaderoch súborov.

V prípade, že chceme povoliť pripojenie pomocou socketu na server, tak nám už nestačí crossdomain.xml, ako v predchádzajúcom prípade. Tentokrát je nutné spustiť Socket Policy File Server. Jedná sa o veľmi jednoduchý server, ktorý má otvorený port 848 a pri požiadavke na tento port odošle naformátovaný XML policy dokument. Ako napríklad:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "/xml/dtds/cross-domain-policy.dtd">
 <!-- Policy file for xmlsocket://socks.example.com -->
<cross-domain-policy> <!-- This is a master-policy file -->
<site-control permitted-cross-domain-policies="master-only"/\>
 <!-- Instead of setting to-ports="*", administrators can use ranges and
     commas -->
<!-- This will allow access to ports 123, 456, 457, and 458 -->
<allow-access-from domain="swf.example.com" to-ports="123,456-458" />
</cross-domain-policy>
```

Port 848 predstavuje pomerne výrazné obmedzenie pre použitie aplikácie hlavne v paranoidnejších sieťach, ktoré povoľujú len malú časť well-known ports.

Ďalšie detaily o tejto problematike nájdete v dokumente:
http://www.adobe.com/devnet/flashplayer/articles/socket_policy_files.html

Zabezpečenie SWF

SWF súbory sú vňčšinou spájané s Flashom. Jedná sa o bajtokód, ktorý je interpretovaný virtuálnym strojom. Podobne je tomu aj v prípade Javy, aj keď virtuálny stroj je odlišný. SWF súbory je možné vytvoriť pomocou nástrojov Adobe ako je Flash Builder, Flash Professionale alebo open source technológií. Špeckácia SWF je otvoreným štandardom. Popis je možné získať tu: http://www.adobe.com/devnet/swf.html

Tak ako .Net alebo Java, tak aj SWF je možné dekompilovať späť z binárnej podoby do zdrojového kódu. To znamená, že sa neodporúča ukladať do SWF žiadne privátne kľúče. Bežný dekompiler je schopný zrekonštruovať dokonca aj stopy po použitom aplikačnom frameworku ako napríklad Swiz alebo Cairngorm.

Do SWF je možné okrem výkonného kódu vložiť aj grafické súbory (assety). Tieto assety je možné tým pádom z SWF pomocou dekompilátora vytiahnuť.

Ochrana pred dekompiláciou býva obfuskácia kódu. Efekt je diskutabilný, pretože útočníka, ktorý začal s dekompiláciou pravdepodobne nezastaví.

Efektívnejšia metóda spočíva v šifrovaní SWF bajtkódu. Túto technológiu dodáva napríklad SimplifiedLogic – riešenie Nitro LM. Jedná sa o cloud riešenie s relatívne nízkou cenou.

Každopádne takáto softvérová ochrana je diskutabilná, pretože nakoniec ju vždy musí zaplatiť platiaci zákazník.

Google Chrome

Google úzko spolupracuje s Adobe. Vrámci prehliadača Google Chrome je dodávaný aj Flash Player. Z tejto spolupráce dvoch firiem vzišlo veľa optimalizácii ako v Google Chrome tak aj vo Flash Playeri. Google Chrome napríklad má implementovaný okolo pluginu (nie len Flash Playeru) bezpečnostný kontajner, ktorý obmedzuje „únik“ škodlivých aplikácii z prehliadača.

Adobe AIR

Jednou z obrovských výhod Adobe Flex technológii je možnosť vytvoriť z webovej verzie, desktop verziu aplikácia a nainštalovať ju ako natívnu. Adobe dodáva technológiu Adobe AIR. Zjednodušene povedané sa jedná o Flash Player pre desktop mimo web prehliadača. Adobe AIR je runtime, tak ako aj Java a umožňuje využiť dostupné funkcie počítača.

Vzhľadom na to, že sa jedná o natívnu aplikáciu, tak odpadajú limitácie s crossdomain, REST a socket serverom. Výhoda AIR runtimu spočíva v tom, že je cross platformný. Podporovaný je Windows a Mac. Linux je podporovaný do verzie AIR 2.6, ďalšie verzie majú byť dodávané partnermi z Open Screen Projectu (http://www.openscreenproject.org/)

Google Android, iPhone a Blackberry

Aplikácie, ktore boli vytvorené pomocou Adobe AIR je možné skompilovať a nasadiť na systém Google Android. Flash Builder umožňuje vytvoriť veľmi jednoducho inštalovateľný APK balíček. Funkcie, ktoré sú aplikácii povolené je nutné špecifikovať v XML manifeste.

Tento istý mechanizmus je možné využiť aj pri kompilácii aplikácií pre iPhone alebo BlackBerry.

Môžete si pozrieť napríklad záznam prednášky z Android DevCampu v Prahe – http://georgik.sinusgear.com/2011/07/19/mobile-first-video-adevcamp-prague/

Flash a HTML5

Flash a HTML5 bývajú mainstreamom označované ako antagonistické technológie. HTML5 so silnou podporou JavaScriptu je skutočne schopné zrealizovať veľa vecí, ktoré Flash Player dokáže. Nadruhú stranu chýbajú často optimalizácie v prehliadačoch pre tieto aplikácie. Adobe už dávno pracovalo na tom, aby sa tieto dve technológie dokázali doplniť. Pomocou ExternalInterface je možné prijímať volania z JavaScriptu, či volať JavaScript funkcie.

Dobrým príkladom je napríklad načítanie súboru pomocou HTML5 File API a jeho spracovanie napríklad pomocou technológie Pixel Bender akcelerovanej na grafickej karte.

Pri použití ExternalInterface sa na požiadavky odoslané cez JavaScript aplikujú bežné bezpečnostné obmedzenia web prehliadačov.

Poznámka na záver

Nástroj pre tvorbu aplikácii postavených na báze open source frameworku Adobe Flex – tzv. Flash Builder je k dispozícii pre študentov a akademických pracovníkov zdarma. http://georgik.sinusgear.com/2010/05/17/
studenti-a-skoly-mozu-ziskat-flash-builder-4-zdarma/

Open source

Na stránkach: http://opensource.adobe.com firmy Adobe nájdete open source aktivity.

Flex framework a informácie súvisiace s integráciou so Spring frameworkom nájdete na: http://flex.org/