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

XXV. konference – XXVth conference

Sborník příspěvků

Conference proceedings

Hotel Klášter
Teplá u Mariánských Lázní
3.–6. 10. 2004

Sborník příspěvků z XXV. konference EurOpen.CZ, 3.–6. 10. 2004

Sazba a grafická úprava: Ing. Miloš Brejcha – Vydavatelský servis
Vytiskl: WOW, s. r. o., Washingtonova 25, Praha 1

ISBN 80-86583-07-4

Obsah

Programový výbor

Roman Pavlík
Jirka Novotný
Martin Macháček
Vladimír Rudolf

Bezpečnost informačního systému organizace – první kroky

Michal Dostál

E-mail: mdostal@kpmg.cz, mdostal@centrum.cz

1 Úvod

Prosazování bezpečnosti v organizaci bude skutečně účinné tehdy, pokud se opírá o správnou identifikaci toho, co chceme chránit – aktiv organizace, a o analýzu rizik, která těmto aktivům hrozí. Nahodilá opatření založená na tom, co nám doporučili výrobci bezpečnostního softwaru, nebo na tom, co jsme odpozorovali od jiných organizací, mohou být plýtváním prostředky nebo mohou dokonce být kontraproduktivní.

V tomto příspěvku se v krátkosti seznámíme se základními pojmy a postupy v oblasti řízení bezpečnosti, jejichž aplikace nám umožní vyvarovat se nejhrubějších chyb při zavádění bezpečnostních opatření.

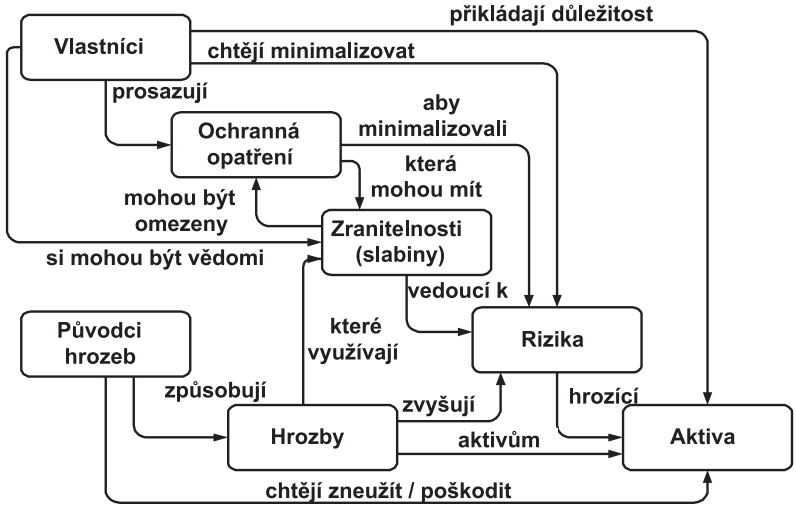
Příspěvek se opírá o informace, obsažené v normách ISO/IEC 17799 a ISO/IEC 13335, část 1 až 4.

2 Analýza rizik

Jedním ze základních nástrojů při určování bezpečnostních požadavků a bezpečnostních opatření je analýza rizik. Ve většině organizací budou poznatky vyplývající z analýzy rizik doplněny hledisky vyplývajícími za zákonů, závazných norem a z cílů organizace. Tato dodatečná hlediska mohou výrazně ovlivnit to, jak organizace vnímá jednotlivá rizika – neplánované hodinové přerušení provozu informačního systému bude mít jiné důsledky v právnické firmě a jiné v bance.

Předmětem informační bezpečnosti je ochrana informačních aktiv před působením rizik a vznikem negativních dopadů, jejichž původcem jsou hrozby využívající slabiny informačních systémů. Ochranná opatření působí na tyto slabiny s cílem je eliminovat, případně zmírnit následky

negativních dopadů. Hrozby mívají své původce, kteří mají svojí motivaci a záměr. Vlastník aktiv má zájem minimalizovat rizika, prosazuje použití ochranných opatření a měl by znát slabiny svého systému.



Předchozí odstavec popisuje základní vztahy mezi prvky, které tvoří koncept informační bezpečnosti a charakterizuje základní cíle prosazování bezpečnosti.

Rizika hrozící informačnímu systému je třeba vyhodnotit, posoudit pravděpodobnost jejich výskytu a potenciální dopad. Tam, kde je to možné, je nejlépe tyto parametry kvantifikovat: Chceme vědět, kolikrát k určitému incidentu pravděpodobně dojde za určité časové období a jaké budou pravděpodobné finanční důsledky dotyčného incidentu. Takto vyhodnocené riziko nám umožňuje posoudit přiměřenost nákladů na protiopatření.

Ve většině případů je třeba počítat s tím, že zvolené protiopatření nevede k plné eliminaci rizika, ale pouze k jeho redukci. Redukované riziko v tomto případě nazýváme zbytkové riziko.

Hlavní činnosti při určování vhodných bezpečnostních opatření jsou:

- identikace aktiv,
- identifikace zranitelností a hrozeb,
- analýza rizik, jejich pravděpodobnosti a dopadu,

- výběr opatření,
- identifikace zbytkových rizik.

Při neformálním intuitivním rozhodování o bezpečnostních opatřeních často některé z těchto kroků alespoň částečně provádíme, i když o tom nepořizujeme formální záznam. Formalizovaný postup při určování vhodných bezpečnostních opatření má ale podstatné výhody. Je snadnější zajistit, že nám neunikne žádná podstatná okolnost, výsledky naší práce může prověřit další osoba, práci na takto pojaté analýze rizik lze rozdělit mezi více lidí. V neposlední řadě je takto formalizovaná analýza přesvědčivějším argumentem pro podporu získání prostředků na ochranná opatření.

2.1 Aktiva

Prvním krokem v rámci analýzy rizik je identifikace toho, co chceme chránit – aktiv, kterými organizace disponuje. Tato aktiva zahrnují:

- fyzická aktiva (hardware, budovy),
- informace a data,
- software,
- schopnost vytvářet určité produkty nebo poskytovat služby,
- lidé,
- nehmotná aktiva (image, vztahy k zákazníkům).

Identifikaci aktiv lze provést na vysoké úrovni. Stupeň podrobnosti by měl být poměřován poměrem času a nákladů vůči hodnotě identifikovaných aktiv.

Informační bezpečnost se zaměřuje především na bezpečnost informačních aktiv. Bezpečnost informací je charakterizována jako zachování:

- důvěrnosti – zajištění toho, aby informace byla dostupná pouze autorizovaným osobám,
- integrity – zajištění toho, že informace nebyla neoprávněně změněna,
- dostupnosti – zajištění toho, aby informace a s nimi spojená aktiva byly k dispozici jejich uživatelům podle potřeby.

2.2 Hrozby a zranitelnosti

Hrozba má potenciální schopnost způsobit nežádoucí incident, vedoucí k poškození aktiv. Základní typy hrozeb jsou:

- Úmyslné lidské hrozby, např. odposlech, neautorizovaná změna informace, krádež.
- Náhodné lidské hrozby, např. chybné vymazání souboru, fyzické nehody.
- Hrozby prostředí, např. požár, povodeň.

Hrozby mají charakteristiky, které nám pomáhají je popsat. Mezi hlavní charakteristiky hrozeb patří:

- zdroj,
- motivace,
- četnost/pravděpodobnost výskytu,
- síla hrozby.

Zranitelnosti jsou slabá místa na úrovni fyzické, organizační, procedurální, hardwaru, či softwaru. Tato slabá místa mohou být využita hrozbami, které mohou vést k poškození aktiv. Naši pozornost zasluhují zranitelnosti, kterým odpovídá určitá hrozba. Analýza zranitelností musí brát v úvahu prostředí a existující ochraná opatření.

2.3 Dopady a rizika

Dopad je důsledek nežádoucího incidentu, který má vliv na aktiva – vede k jejich zničení či poškození. Kvantifikace dopadů umožňuje vytvoření rovnováhy mezi náklady na ochraná opatření a potenciálními důsledky nežádoucích incidentů. Přitom je třeba přihlížet k četnosti výskytu nežádoucího incidentu.

Riziko je potenciální možnost, že určitá hrozba využije zranitelnosti, aby způsobila ztrátu nebo poškození aktiv. Závažnost rizika je charakterizována kombinací dvou faktorů, pravděpodobnosti výskytu nežádoucího incidentu a jeho dopadu.

2.4 Ochranná opatření

Ochraná opatření jsou akce, které mohou poskytnout ochranu před hrozbou, snížit zranitelnost, omezit dopad incidentu, detekovat nežádoucí incidenty nebo usnadnit obnovu. Ochranná opatření mohou být použita v oblastech fyzického prostředí, technického prostředí (hardware, software, komunikace), personálu a administrativy. Ochranná opatření, která byla zvolena na základě analýzy rizik, tedy ochranná opatření, která chrání identifikovaná aktiva před reálnými hrozbami, budou podstatně účinnější než ochranná opatření vybraná nahodile nebo intuitivně.

2.5 Zbytkové riziko

Ochranná opatření typicky dosáhnou redukce rizika, nikoliv jeho plné eliminace – mimo jiné s ohledem na rovnováhu nákladů mezi ochrannými opatřeními a potenciálními důsledky incidentů. Je důležité, aby takto vzniklá zbytková rizika byla identifikována a akceptována vedením organizace, které je oprávněno akceptovat dopad výskytu těchto incidentů, případně autorizovat použití prostředků na dodatečná ochraná opatření.

3 Řízení bezpečnosti

Předchozí část příspěvku se zaměřila na jednorázovou analýzu rizik a určení přiměřených ochranných opatření. Prosazení bezpečnosti v organizaci ale není jednorázovou událostí, nýbrž trvalým procesem. Obrázek získaný analýzou rizik musí nutně zastarat. Kromě toho je nutno řízení bezpečnosti zahrnout do organizační struktury organizace – někdo musí tato opatření navrhnout, schválit, alokovat potřebné zdroje, prosadit je a kontrolovat jejich účinnost. V neposlední řadě je zapotřebí řízení bezpečnosti dokumentovat.

Mezi hlavní požadavky na systém řízení bezpečnosti patří

- koordinace aktivit, sloužících k prosazení bezpečnosti,
- alokace zdrojů,
- určení zodpovědnosti a pravomoci,
- revize a hodnocení bezpečnostních opatření.

Řízení bezpečnosti zahrnuje soubor klíčových procesů, mezi které patří:

- analýza rizik,
- tvorba a údržba bezpečnostní politiky,
- klasifikace a kontrola aktiv,
- personální bezpečnost,
- reakce na incidenty,
- fyzická bezpečnost,
- údržba dokumentace,
- řízení změn,
- řízení přístupu,
- řízení konfigurací,
- vývoj a údržba systémů,
- řízení kontinuity,
- soulad s právními a dalšími požadavky.

Systém řízení bezpečnosti určuje zodpovědnosti a pravomoci tak, aby bylo zajištěno fungování zmíněných procesů.

Zásadním krokem k formalizaci systému řízení bezpečnosti je formulace *Bezpečnostní politiky informací*. Tento dokument definuje přístup organizace k informační bezpečnosti a vyjadřuje podporu vedení organizace. Jsou jí podřízeny další směrnice, procedury a bezpečnostní politiky jednotlivých systémů.

Dokument *Bezpečnostní politika informací* typicky obsahuje:

- definici informační bezpečnosti, její cíle a důležitost,
- prohlášení vedení organizace o záměru podporovat informační bezpečnost,
- stručnou formulaci bezpečnostních zásad a principů,

- stanovení odpovědnosti za jednotlivé činnosti vedoucí k prosazování bezpečnosti,
- odkazy na detailnější dokumentaci.

Aby bezpečnostní politika zůstala účinným nástrojem k prosazování bezpečnosti, je zapotřebí ji udržovat a aktualizovat. Aktualizace by měly reagovat na změny ovlivňující východiska původního hodnocení rizik a stanovení bezpečnostních požadavků. Bezpečnostní politika může být aktualizována i v rámci reakce na incidenty, nové zranitelnosti, změny v organizaci a nové technologie.

4 Závěr

Účinnou metodou určení přiměřených bezpečnostních opatření je analýza rizik, při které identifikujeme aktiva, která chceme chránit, a hrozby, před kterými je chceme chránit. To nám umožní porozumět rizikům a jejich možným důsledkům a zvolit přiměřená protiopatření. Analýza rizik je jedním z hlavních procesů systému řízení bezpečnosti. Správně nastavený systém řízení bezpečnosti přispěje k udržování a zlepšování úrovně bezpečnosti organizace.

Literatura

[1] ISO/IEC 17799 Information security management.

[2] ISO/IEC TR 13335 Information technology – Guidelines for the management of IT Security.

[3] ISO/IEC 15408-1 Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model.

Základní pravidla pro tvorbu bezpečných systémů Možnosti a omezení bezpečnosti v ICT systémech

Jan Müller

E-mail: jan.muller@i.cz

Vaine est l'illusion des sédentaires qui croient pouvoir habiter en paix leur demeure car toute demeure est menacée.
Mýlí se lenoši, kteří se domnívají, že mohou netečně obývat svůj dům, neboť každé obydlí je neustále ohroženo.

Antoine de Saint-Exupéry

1 Úvod

V poslední době jsme svědky toho, jak v důsledku explozivního růstu využívání informačních a komunikačních technologií dochází také k zásadnímu růstu významu bezpečnosti aktiv, zpřístupňovaných a zpracovávaných technologiemi ICT, a to ve všech základních charakteristikách jako je integrita, důvěrnost, dostupnost apod. Změna paradigmatu ICT měla dramatické dopady v různých oblastech managementu rizik:

- principiálním způsobem se rozšířil rozsah a význam dat dostupných přes ICT (tzn. vzrostla hodnota aktiv);
- manipulace těchto aktiv prostřednictvím ICT (např. prostřednictvím Internetu) je pro stále větší okruh lidí stále běžnější, tj. otevřela se řada nových možností přístupu k aktivům (tzn. narostly hrozby vůči sledovaným aktivům);
- ve stále komplexnějším prostředí uživatelé k těmto manipulacím používají laickým způsobem nástroje, kterým nerozumí a jejichž hlavní návrhová kritéria byla obchodní úspěšnost a ne bezpečnost (tzn. zvýšil se počet zranitelností);

- průniky do interních sítí nebo kompromitování přenášených dat jsou i díky automatizovaným nástrojům a nezabezpečenému programovému vybavení nyní mnohem běžnější a efektivnější jak pro zábavu a prestiž, tak i v rámci obchodního boje a čím dál tím víc i v rámci běžné kriminality (tzn. zvyšují se rizika).

Tím, jak bezpečnostní problematika nabývá na významu, se také stále více ukazují její dvě základní vlastnosti; na jedné straně bezpečnost zahrnuje obrovské spektrum velmi nesourodých oblastí, od pistolníků ve dveřích až po kryptografickou analýzu postranních kanálů, a na druhé straně mohou být obrovské investice do bezpečnosti anulovány selháním jediného poměrně nevýznamného článku – celková bezpečnost je jenom tak dobrá, jak je dobrý nejslabší článek v celém uvedeném spektru, od pistolníků až po postranní kanály.

Řešení bezpečnosti v ICT si více než cokoliv jiného můžeme představit jako komisi slepců (tj. odborníků v jednotlivých oblastech :–), kteří se pokoušejí ohmatáváním jednotlivých částí slona popsat celého slona a podle svého zaměření spatřují řešení např. v zavedení ISMS (Information Security Management System) podle standardu ISO/IEC 17799, zakoupení konkrétního SW jako např. firewall od Checkpointu, zahození konkrétního SW jako např. IIS od MS, korektní konfigurace konkrétního SW jako je firewall nebo web server, nasazení silných proaktivních technik (IPS), nasazení silných forenzních analytických technik (post mortem analysis), použití silné kryptografie nebo přímo konkrétně PKI, prosazení procesní kontroly tvorby SW a jejich bezpečnostni testování podle nějakých metodologií nebo profilů (např. ISO/IEC 15408), použití hardwareových tokenů, automatické aktualizace SW a ověřování jejich autenticity, výstražné bičování usvědčených hackerů atd. atd.

Navrhovaných technologií a postupů jsou tisíce, každá z nich obvykle víceméně řeší některý aspekt bezpečnosti, žádná ovšem ne absolutním způsobem (tj. i v dané omezené oblasti může dojít k selhání), a žádná ovšem nemůže řešit bezpečnost v celé šíři. Smutné ovšem je, že ani při komplexním nasazení různých technik nemůžeme mít skutečnou jistotu bezpečnosti ...

Má tedy vůbec smysl se bezpečností zabývat, jaký přínos získává uživatel za vloženou práci a investice, čeho lze reálně dosáhnout a na jakých externích okolnostech dosažené výsledky závisí?

Předkládaný příspěvek konstatuje, že obecně nelze očekávat všelék na bezpečnost, na druhé straně ovšem je možné zásadním způsobem zlep-

šit bezpečnostní situaci v daném systému, pokud budeme při návrhu analýzy a řešení postupovat systémově, pokud budeme rozumět funkcionalitě objektového systému a prostředí, ve kterém je tento systém provozován, pokud budeme chápat to, že dosažitelné cíle jsou omezeny (a tato omezení přijmeme) a zejména pokud pochopíme, že bezpečnost není „feature“, ale trvalý proces.

Příspěvek si proto neklade si za cíl nabízet řešení, ale spíše se zabývá celkovým popisem problematiky bezpečnosti, popisem některých významných oblastí, jejich možnostmi a omezeními, a zejména popisem jejich provázanosti a interakcí. Uvedené pohledy vycházejí z empirických zkušeností z různých vzdálených oblastí ICT a snaží se popsat komplexní pohled na celkovou bezpečnost objektových systémů.

2 Komplexnost jako základní charakteristika ICT

> *One of the most important lessons we can all learn about Critical Information Infrastructure Protection can be summarised in one word – interdependency.*
>
> Stephen Cummings
> Director NISCC
> *Security's worst enemy is complexity.*
> *A Cryptographic Evaluation of IPsec, Ferguson, Schneier*

Jedním ze základních důsledků informační revoluce je i fakt, že se ICT technologie a jejich infrastruktura stává součástí tzv. Kritické infrastruktury – tím, že ICT infrastruktura zabezpečuje manipulaci s klíčovými aktivy, se tato infrastruktura stává sama klíčovým aktivem.

Ve skutečnosti ovšem tyto procesy obecně odrážejí pouze změny v přístupu k CIP (Critical Infrastructure Protection) a zde není pochyb o tom, že ICT technologie, služby a infrastruktura se stávají součástí kritické infrastruktury a že na jejich dostupnosti a integritě závisí hladký chod společnosti stejně jako na dodávce pitné vody.Na druhé straně se ukazuje, že bezpečnost kritických systémů, služeb a infrastruktur je nyní nesrovnatelně složitější zejména vzhledem ke kritické závislosti klíčových komponent na řadě dalších, ne vždy přesně definovaných systémů, a ICT technologie a systémy jsou právě tím faktorem, které umožňují, aby tyto vzájemné závislosti dosáhly kritické úrovně. Po září 2001 se zásadně změnil pohled většiny států na CIP, a nový přístup je charakterizován právě

důrazem na ICT a na vzájemnou závislost jednotlivých sektorů. Tato nová koncepce vyústila v řadu specifických programů, které se těmito aspekty zabývají. V Holandsku se např. program QuickScan soustředil právě na zmapování závislostí a kaskádových efektů, a naopak cvičení Livewire v USA, pořádané ministerstvem DHS, které simulovalo útoky na kritickou infrastrukturu ukázalo, že největší problémy stále přetrvávají v provázaností jednotlivých systémů, ve schopnostech komunikace apod.

Komplexnost a provázanost dat a informačních služeb nad těmito daty tedy představuje fundamentální paradigma informační společnosti a současně představuje i nejvýznamnější problém bezpečnosti ICT systémů.

Z pohledu vlastního řešení bezpečnosti ICT je ovšem třeba rozlišit komplexnost v několika různých rovinách:

- komplexnost vlastního prostředí a příslušných procesů,
- komplexnost ICT řešení (aplikační programy sledující komfort uživatele),
- komplexnost vlastních bezpečnostních řešení.

Zatímco komplexnost prostředí a příslušných procesů je daná objektivní realitou, komplexnost ICT řešení nebo bezpečnostních řešení je spíše důsledkem přístupu systémového architekta tohoto řešení. Technické řešení pak může být ovlivňováno externími požadavky na funkcionalitu (backward compatibility) nebo snahou o to, aby se uživateli poskytlo co nejvíce funkcí a uživatelské přítulnosti, aniž by s tím byl zatěžován – uživatel se pak často nestačí divit, co všechno je na jeho počítači naistalováno v otevřených defaultních konfiguracích a jak samostatně tyto programy komunikují s jakýmisi externími entitami. Řada dalších problémů při návrhu pak může být způsobena obchodními prioritami výrobce, kde bezpečnost není primárním cílem („throwing features into their products and dealing with the problems later“), a nakonec se často objevují i problémy způsobené tím, že návrh architektury řešení je příliš technologicky zaměřen a reflektuje zkušenost řešitelů z jiných, neaplikovatelných problémových oblastí. Zcela stranou pak ponecháme fakt, že komplexnost může být i cíleným obchodním trikem se záměrem ztížit konkurenci a uživatelům přístup do proprietárního systému.

Typickým příkladem příliš komplexního objektového systému mohou být konkrétní balíky vzájemně nesystémově provázaného kancelářského SW a jejich vazby do operačního systému – Code Red například mohl napadat i ty instalace Windows, které nesloužily jako webové servery, protože celé balíky služeb byly vzájemně propojené a IIS byl potajmu instalován, aniž by si to uživatel explicitně přál. Obdobné přístupy a následné problémy ovšem můžeme potkat i na nejnižší úrovni v hardwareových architekturách (společný stack Intelu) nebo naopak v celých velkých síťových a aplikačních architekturách (Komunikační Infrastruktura Veřejné Správy).

Nesystémový přístup a následnou komplexnost ale můžeme však najít i v celých (standardizovaných) protokolových systémech, např. protokol Link16, používaný v armádních systémech NATO, který obsahuje aplikační data v záhlaví paketů linkové vrstvy, nebo třeba celý protokolový stack SS7 (signalizační protokol mezinárodně používaný na úrovni trunků mezi telefonními ústřednami), což je jeden nestrukturovaný moloch (Tanenbaum: „gigantic unstructured mess“), do kterého byly částečně dolepeny některé novější moduly (TCAP/SCCP pro překlady čísel).

Produkty nebo systémy tohoto typu se pak dají jen obtížně zabezpečit, mj. i proto, že nelze provést adekvátní rizikovou analýzu, a často pak mohou fungovat přiměřeně bezpečně pouze v chráněném prostředí.

Ještě horší je situace v případech, kdy nevhodný návrh vede k příliš komplexnímu řešení vlastních bezpečnostních opatření, kdy pak paradoxně implementace bezpečnostních opatření může negativně ovlivnit celkovou bezpečnost systému; i zde se tyto problémy mohou týkat konkrétních produktů, celých systémů nebo i protokolových stacků.

Příkladem takových produktů mohou být univerzální všeho schopné firewally, které implementují kromě funkcí BPS (Boundary Protection Services) také celou řadu dalších služeb (AV, routing, traffic shaping apod.) a nejsou schopny jednotlivé funkce oddělit a delegovat. Uživateli, ale mnohdy ani výrobci, nemusí pak být zřejmé, jestli například při přetížení nebo selhání některého subsystému nezůstane firewall otevřený (fail-open).

U protokolových stacků je zářným příkladem protokol IPSEC (RFC 2406), který je v mnoha ohledech typickým výsledkem práce komise – příliš mnoho možností ve snaze zalíbit se všem, zbytečné použití IP adresy pro identifikaci SA, příliš mnoho variant a překrývající se funkciona-

lity na úrovni modů (transport a tunnel) nebo vlastních protokolů (AH a ESP) atd. Řada problémů pak existuje nejen na úrovni bezpečnostní (viz „Cryptographic Evaluation of IPsec“, Ferguson, Schneier), ale i na úrovni čistě technické, např. při použití NAT (viz RFC 3027 Protocol Complications with the IP Network Address Translator).

U takto složitých bezpečnostních řešení a univerzálních produktů pak je hlavním problémem zajištění potřebné úrovně důvěry v implementované řešení (assurance ve smyslu Common Criteria ISO/IEC 15408). Ostatně je typické, že bezpečnostní profily PP (Protection Profile podle CC), vydávané Ministerstvem Obrany U.S.A. pro oblast firewallů, specifikují v popisu cílového systému TOE (Target of Evalution) velmi omezenou funkcionalitu, aby vůbec bylo možno dosáhnout potřebné třídy EAL (Evaluation Assurance Level).

3 Kritické oblasti bezpečnosti

V této kapitole následuje několik odstavců, které poměrně nesystematicky popisují některé vybrané oblasti bezpečnosti ICT (na způsob prof. Jedličky, který podle prof. Šikla „rozděluje pneumonie jako klobouky na velké, zelené a filcové“).

3.1 Systémový pohled – ISMS podle BS7799 a řízená implementace

Jestli toto všechno budeš dělat, dobře se ti povede.

7 trpaslíků Sněhurce

Pokud by se mělo interní IT organizace soustředit na jeden aspekt a jeden přístup k bezpečnosti, mělo by to být právě řízení bezpečnosti podle BS 7799/ISO 17799 (v rámci tohoto sborníku se těmto základním standardům věnují jiné příspěvky). Všechny podstatné oblasti jsou zde podchyceny a i malé organizace budou těžit z konkrétních definic odpovědností za jednotlivé oblasti a za jasné definování procesů alespoň na rudimentární úrovni; mimořádně důležitý je i fakt, že postup v souladu se standardem zajišťuje, že pokud byla některá ze základních oblastí (jedna z 10 tzv. control clauses) nebo některé z jejich cílů (jedna z 36 tzv. control objectives) vynechána nebo extrémně zjednodušena, nestalo se tak nedopatřením, ale záměrně. Jestliže tedy organizace nemá např.

BCP (Business Continuity Plan), pak ho nemá záměrně a rizika spjatá s tisíciletou vodou a zemětřesením bere na sebe.

Za normálních okolností by pak management neměl být schopen setřást odpovědnost v tom smyslu, že potřeba řešit danou oblast bezpečnosti byla jasně deklarována a rovněž tak by měl existovat člověk, který je za to přímo odpovědný. Standard je ostatně procesně orientován a jeho základní části (ISO 17799:2000 Part 1) jsou harmonizovány s ISO 9000.

BP a ISMS nejsou samoúčelné, cílem je redukovat rizika na akceptovatelnou míru (reziduální riziko). Podobně jako u dalších aspektů bezpečnosti, ISMS je ale jen jednou částí celkové bezpečnosti, která sama o sobě nemusí postačovat k zabezpečení ICT. Procesní rámec typu ISMS totiž představuje nutnou podmínku, která popisuje, jaká opatření by měla být provedena, ale nevypovídá nic o jejich kvalitě. Z tohoto pohledu je tedy zavedení řízení bezpečnosti podle ISO 17799 analogické zakoupení firewallu – pokud nebude správně konfigurován, provozován a udržován, pak se jedná jen o mrtvou investici. Ani ISMS tedy nepředstavuje univerzální lék na všechno.

Tam kde je splnění některých bezpečnostních požadavků požadováno dalšími formálními mechanismy jako jsou např. audity pro výroční zprávy v USA, svádí navíc formalizované metodologie k tomu, aby naplnění těchto požadavků bylo zajištěno pouze čistě formálně.

Rik Farrow tak např. v časopisu *;login:* kdysi vyprávěl apokalyptickou historku o tzv. checkbox firewallech, které slouží jen k tomu, aby si je auditoři odškrtli a v některých konkrétních případech byl takto odškrtnutý firewall ještě uložen v originální krabici ...

3.2 Možnosti uživatelů v bezpečnosti SW – skepticismus a obrana do hloubky

„Oracle 9i. Unbreakable. Can't break it. Can't break in."

Reklamní kampaň Oracle v r. 2002

Řada výrobců se v současné době halasně ohání bezpečností svých produktů, protože to je evidentně obchodně úspěšná strategie, ale není vůbec zřejmé, do jaké míry tyto produkty skutečně bezpečné jsou. Uvedu zde dva příklady velmi přesvědčivé kampaně dvou renomovaných výrobců, Oracle a Microsoft, kde celé kampaně byly založeny na zcela nové vysoké úrovni bezpečnosti jejich produktů, a v obou případech se záhy ukázal opak.

Normální uživatel ovšem nemá zdroje na to, aby se o reálné úrovni bezpečnosti vůbec něco dozvěděl, a je poměrně běžné, že v tom nemá moc jasno ani výrobce – bezpečnost se nedá testovat jako třeba počet transakcí za vteřinu a to, že tisíc útoků neprojde neznamená, že neuspěje tisící prvý. Kromě toho si sice někteří výrobci skutečně upřímně (a extrémně naivně) myslí, že jejich produkt je bezpečný (to byl zřejmě případ zmíněné reklamní kampaně Oracle), ale jiní si s tím zdaleka tolik hlavu nelámou.

Co tedy může uživatel dělat, aby vyhodnotil bezpečnost kupovaných produktů v současné době komplexního aplikačního prostředí a komplexních softwareových produktů?

Evidentně jako uživatelé potřebujeme nezávislé bezpečnostní testy a ohodnocení nejrůznějších ICT produktů a služeb, ale i zde je třeba postupovat opatrně a to i přesto, že některé vyhodnocovací metodiky jsou velmi sofistikované. V tomto směru je velmi poučný případ reklamní kampaně Oracle 9i, který byl ještě téhož roku hacknutý několika exploity s využitím triviálních buffer overflow. Oracle pak tvrdil, že unbreakable znamená, že produkt prošel 14 nezávislými bezpečnostními evaluacemi (bylo jich ve skutečnosti jenom pět, protože Oracle počítal různé úrovně jako samostatné evaluace ;–). Přesto se ale nejednalo o nějaké amatérské testy (byly tam např. TCSEC, ITSEC, CC, FIPS 140-1), ale ani ty nedokázaly odhalit buffer overflow. Znovu se tedy naskýtá otázka, že pokud problémy (jako je poměrně triviální buffer-overflow) nedokázal odhalit Oracle s použitím oficiálních metodologií, jakou šanci má spotřebitel se v této problematice vůbec orientovat?

Podobně koncem roku 2001 prohlásil viceprezident Microsoftu Jim Allchin:
Windows XP is dramatically more secure than Windows 2000 or any of the prior systems. Buffer overflow has been one of the attacks frequently used on the Internet. We have gone through all code and, in an automated way, found places where there could be buffer overflow, and those have been removed in Windows XP. We have also turned off by default a whole set of things so that users are configured in a minimalist kind of way, making them less vulnerable.

K tomu poznamenal Bruce Schneier, že si tyto citáty rád schovává a že se k nim do roka vrátíme. Ale již v únoru 2002 napsal v Cryptogramu:

Anyone remember last November, when Microsoft VP Jim Allchin said in an eWeek interview that all buffer overflows were eliminated in Windows XP? Or that it installed in a minimalist way, with features turned off by default? Not only was the UPnP vulnerability in an unneeded feature that was enabled by default; it was a buffer overflow.

Ukazuje se tedy, že podlehnout marketingovým kampaním může být pro bezpečnost vaších systémů velmi nezdravé, a to ne proto, že v programu jsou chyby (v programu jsou VŽDY chyby), ale proto, že jste se na to zcela spolehli.

V některých případech to dokonce ani nemusí být rouhavé spoléhání na celkovou bezpečnost systému, ale prosté používání některé standardní služby. Kdo si např. myslel, že má soubor ve Wordu zabezpečen použitím hesla, byl asi překvapen, když letos na jaře Microsoft prohlásil, že toto heslo není „security feature“ a že uživatelé mají použít nějaké kryptografické programy (a to pochopitelně proto, že šikovní hoši ukázali, že heslo se dá prostě obejít s pomocí hex editoru).

Vzhledem ke komplexnosti problematiky se tedy uživatel nemůže sám orientovat a evidentně se nemůže spoléhat na prohlášení výrobců. Na výrobce je samozřejmě nutno vyvíjet tlak, aby změnili svůj postoj k bezpečnosti (konec konců v Microsoftu už k jistým posunům došlo) a také specificky aby podrobovali své výrobky různým bezpečnostním auditům. Jak je ale vidět, ani na to nelze zcela spoléhat.

Za těchto okolností se uživatelé musí věnovat ochraně svých systémů také s použitím selského rozumu, a zde patří mezi nejdůležitější principy **ochrana do hloubky**. V žádném případě by tedy nemělo dojít k situaci, kdy selhání jedné komponenty obrany, typicky prvku „perimeter defense“ jako je firewall, nechá celý systém zdrojů organizace nechráněný. Mezi další prvky ochrany patří pochopitelně šifrování dat (včetně záloh), ale také opatření na úrovni aplikací jako jsou silné interní autentizace nebo auditní mechanismy, opatření na úrovni sítí jako je přepínání, mikrosegmentace nebo šifrované VPN, opatření na úrovni operačních systémů, zejména v kernelu (Posix1.e, Domain and Type Enforcement, různé security kernel patche jako LSM apod.), ale také obecné zvýšení odolnosti (hardening), různé typy IDS, logování atd. atd. Zcela obecným principem je zde také určitá modularizace a oddělení (kompartmentalizace?) zdrojů tak, aby kompromitování jedné části zdrojů neohrozilo ostatní.

K dalším podstatným principům patří trvalý dohled nad systémy tak, aby případné průniky nebo DoS útoky byly zaregistrovány co nejdříve a aby se tak snížila expozice zdrojů organizace.

3.3 Provoz, údržba a sledování jednotlivých systémů a jejich aktualizace

Be quick, or be dead.

Iron Maiden Prevention is ideal, detection is mandatory.

Je zřejmé, že s informační revolucí narostly nejen objemy a důležitost dat manipulovaných ICT technologiemi, ale také narostl počet útoků, jejich závažnost a efektivnost. To souvisí i s tím, že se značně zvýšil i počet zranitelností – podle statistik CERTu bylo v roce 1995 hlášeno 171 zranitelností, v roce 1999 417 a v roce 2001 již 2 437, za první dva kvartály 2004 zatím 1 740 zranitelností.

Jedním z důsledků tohoto faktu pro uživatele je i to, že organizace musí věnovat stále větší objem kvalifikovaných zdrojů na zabezpečování svých systémů. To se obecně skládá z aktualizací programového vybavení na serverech a klientských stanicích, ale dále také z aktualizovaní a provozování vlastních bezpečnostních mechanismů jako jsou firewally, IDS apod. (samostatným subsystémem je dále správa certifikátů, údržba CRL aj.). Zde je nutno zdůraznit, že u těchto systémů nestačí je zakoupit a dokonce ani správně zkonfigurovat, ale kromě průběžných aktualizací je nutno je také trvale dohledovat a provozovat. Obecně totiž neplatí, že systém se ubrání do nekonečna sám, tzn. je zapotřebí lidský zásah, a primárním cílem je reagovat na probíhající útok co nejdřív. B. Schneier kdysi uváděl příklad cejchování (rating) sejfů, které jsou označeny např. 30TL (30 minutes, tools) nebo 60 TRTL (60 minutes, torch and tools). Taková pokladna není „nedobytná“, ale ten rating prostě říká, že profesionál, vybavený nástroji a acetylénovým hořákem ji otevře během hodiny, „a jestliže nezazní alarm a stráže během té hodiny nepřiběhnou, pokladna je k ničemu“. Mnoho útoků v ICT má podobný charakter a např. SQL Injection, často používaný proti databázím za webovým serverem, obvykle vyžaduje delší zkoumání reakcí databáze a různé přípravné práce.

Celá tato oblast je tedy poměrně složitá a zahrnuje nejen správné rozmístění různých NIDS, HIDS, IPS apod., ale především vyhodnoco-

vání jejich výstupů (pozor na falešná pozitiva!) a také vazby na interní mechanismy incident response.

Pokud jde o ochranu proti nejnovějším exploitům a virům, je zřejmé, že jedinou alespoň částečnou ochranu je okamžitá aplikace bezpečnostního patche programového vybavení nebo aktualizace signatur AV systému či IDS databáze. Na Internetu se znovu a znovu objevují úspěšné exploity, využívající chyb, které byly objeveny v minulém tisíciletí, a přesto napadené systémy neměly aktualizované programové vybavení. Obvyklá reakce je, že útok tedy uspěl jen díky neschopnosti příslušných administrátorů; to je sice víceméně pravda, ale současně je třeba si uvědomit, že s narůstajícím počtem zranitelností, ale také s narůstajícím počtem dalších úkolů okolo bezpečnosti se pro administrátory stává včasná aktualizace všech systémů organizace velmi problematickou. Zdá se tedy, že jediným dlouhodobým řešením je automatizace updatů stejně jako se automatizovala třeba správa uživatelů nebo jejich programového vybavení. Přestože to je asi nakonec jediné rozumné řešení, může mít automatizace celou řadu háčků, a i zde je třeba použít selský rozum, protože ani podepisované patche a updaty nejsou všelék. Nálepka AOC na láhvi francouzského vína mi asi zaručí, že víno skutečně pochází z dané oblasti, nicméně víno nemusí být dobré, a podobně patch podepsaný Microsoftem je asi skutečně od nich, ale může mi spolehlivě sestřelit provozní server.

3.4 Kryptografie

> *If you think cryptography will solve the problem, then you don't understand cryptography and you don't understand your problem.*
>
> SANS Newsbites, unknown

V průřezovém článku o bezpečnosti se nelze nezmínit o kryptografii, která přestavuje jednu z klíčových opor bezpečnosti ICT. Na druhé straně se jedná o tak rozsáhlou a složitou oblast, že zde může jít pouze o pár poznámek – ty se budou týkat především postojů a pověr okolo kryptografických technologií, širšího zakotvení těchto technologií v ICT bezpečnosti a také jejich možnosti a omezení.

Hned v úvodu je třeba říci, že problémy v kryptografii jsou jednak v samotné oblasti kryptografie (zde jde především o různé algoritmy a tedy v zásadě o matematickou problematiku), a potom o problémy

využití kryptografie, které se opět dělí na problémy kryptografických nástrojů a na problémy toho, jak aplikace tyto kryptografické nástroje využívají. Obecně totiž platí, že uživatelé mají tendenci být ohromeni a uspokojeni tím, že aplikace jejich data zašifrovala, a obvykle si neuvědomují, že použitý algoritmus může být tak slabý, že jejich zašifrování je bezcenné. Ještě méně si ovšem uvědomují, že použitý algoritmus může být třeba AES, ale jejich zašifrování může být opět bezcenné, protože implementace algoritmu v knihovně, ale ještě častěji využití tohoto nástroje aplikací, je chybné. Chyb může být samozřejmě mnohem víc, může se jednat o chybné uložení dat v proprietárním formátu aplikace (keyrings v PGP), o chyby ve standardním kódování dat (ASN.1), šifrování může přidávat známý text (standardizovaná záhlaví) a tím usnadnit útoky typu known-text atd. Šifrování má navíc plnit konkrétní úkol, a použité technologie musí být adekvátní – jestliže například banka používá symetrický klíč jak pro šifrování komunikace se zákazníkem, tak i pro jeho autentizaci, pak příslušnému úředníkovi v principu nic nebrání v tom, aby s tímto klíčem autorizoval převod se zákazníkova konta na své.

Kryptografie tedy neexistuje ve vzduchoprázdnu a k jejímu úspěšnému nasazení je potřebujeme jak efektivní a bezpečné algoritmy, tak i jejich korektní implementace, ale také jejich korektní používání. Specifickým příkladem je PKI, tedy Public Key Infrastructure; rychlejšímu rozšíření stále brání nejen to, že se jedná o koncepčně složitou záležitost, ale také fakt, že většina pozornosti se upřela na korektní nástroje, a zapomnělo se na zbytek. Vybudování skutečné infrastruktury je poměrně mnoho práce, protože nezahrnuje jenom technologickou část jako různé servery a databáze, úložiště či OCSP respondéry, ale zejména také pravidla pro vydávání certifikátů, ověřování identit apod. – „PKI is 10 percent technology and 90 percent policies and procedures“. I po vybudování PKI máme ovšem jenom infrastrukturu, skutečná práce může teprve začít při překlápění starších, nedokumentovaných a proprietárních aplikačních programů organizace tak, aby k autentizaci skutečně využívali prostředky PKI.

Zcela obecným problémem kryptografie tedy je fakt, že ačkoliv všichni souhlasí s tím, že celý řetěz je silný jen jako nejslabší článek, tak přesto mají jakousi mystickou víru, že silný kryptografický prostředek zachrání i zbytek řetězce.

Jako příklad problematického okolí kryptografických nástrojů lze upozornit na to, že prostředek, kde se provádějí citlivé kryptografické

operace jako je např. načtení passphrase a vybalení privátního klíče, je obvykle univerzální počítač s „discretionary access control“ a s programovým vybavením orientovaným na pohodlí uživatele a ne na bezpečnost. Bez ohledu na sílu algoritmu zde pak není problém kompromitovat passphrase nebo i privátní klíč, a dokonce i v případě, že se toto všechno odehrává na čipové kartě (i kdyby měl snímač vlastní klávesnici na passphrase/PIN), tak je stále možno např. modifikovat podepisovaný text – uživateli předložím na obrazovce jeden text smlouvy a nechám ho podepsat jiný. Pokud tedy není možno ve stroji vybudovat chráněnou cestu (protected path), což by obvykle vyžadovalo multi-level security podle TCSEC, tak je integrita celé operace ohrožena především prostředím, ve kterém se provádí (viz např. NSA paper: „The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environment“).

I zde je tedy uživatel v nešťastné pozici, kdy musí pro zajištění bezpečnosti používat technologie, kterým nerozumí, a kde není schopen ověřit jejich korektnost. I zde je třeba se spoléhat na ověřená řešení (například se vyhnout proprietárním algoritmům), dávat přednost jednoduchým řešením, kde je negativní vliv komplexních systémů minimalizován, ale také třeba sledovat zprávy o bezpečnostních mezerách programového vybavení (viz např. v minulosti chyby v PGP, SSL nebo SSH).

3.5 Vzdělávání a změna postojů managementu, návrhářů a uživatelů

Bezpečnost je opruz ...

Klíčovým segmentem populace, který je třeba vychovávat k bezpečnosti, je pochopitelně management, který drží v rukou pytlik s penězi a bez jehož podpory a pochopení se bezpečnost ICT v organizaci nemůže ani rozjet. Je příznačné, že svrchu zmiňovaný standard ISO 17799 věnuje poměrně značnou část úsilí při zavádění systémů řízení bezpečnosti právě spolupráci s nejvyšším managementem, ale je třeba zdůraznit, že zde musí odpovědný technický management počítat s tím, že to pro ně bude mnoho náročné práce. Jedna z věcí, kterou musí management pochopit, je také fakt, že řešení bezpečnosti je podstatně složitější než nákup té pravé technologie:

SANS: A KPMG survey of 500 executives from multinational corporations found that the majority believed the solution to security problems is to purchase the right technology. KPMG says they are wrong and that developing a strategy that includes education, training and policy is a more effective response.

Pro většinu aktérů v ICT je bezpečnost něco, co jim znesnadňuje život – uživatel musí procházet různými autentizačními a autorizačními kontrolami, zadat heslo nebo passphrase, absolvovat různé administrativní procedury, generovat klíče, žádat o certifikát atd., a jeho možnosti jsou navíc na každém kroku omezovány. Podobně ale i implementátor naráží na řadu problémů, které mu ztěžují život; v tomto případě navíc musí jednak sám integrovat rozhraní do různých knihoven pro ověření autentizace/autorizace (případně komunikovat s AAA servery), ale také musí sám produkovat bezpečný a auditovatelný kód. Zdaleka se přitom nejedná jen o oblíbené buffer overflows, které představují poměrně jednoduchý problém na úrovni implemetace, ale jde také o serioznější problémy na úrovni návrhu, jako např. návrh bezpečného sandboxového mechanismu a obecně separace práv, zajištění mezi-procesové komunikace, synchronizace apod.

V některých případech se setkáváme i s cílenou snahou, jak obcházet bezpečnostní mechanismy jako je firewall i ze strany systémových návrhářů, nejen ze strany hackerů (jistě umí kdekdo protlačit data přes firewall pomocí třeba DNS).
Microsoft documentation: „Since SOAP relies on HTTP as the transport mechanism, and most firewalls allow HTTP to pass through, you'll have no problem invoking SOAP endpoints from either side of the firewall.“

Pochopitelně situace se tady v mnoha ohledech změnila, SOAP je normální nástroj a značně se změnily i možnosti firewallů, ale bez ohledu na to je snaha návrháře obejít bezpečnostní politiku, prosazovanou na firewallu systémovým administrátorem, fundamentálně chybná.

Cílem není si zjednodušit život, ale realizovat požadovanou funkčnost za dodržení všech dalších specifikací (mj. i bezpečnostních). Pro dosažení korektního postoje návrhářů a implementátorů je proto nutno vyvinout trvalý tlak a vzdělávat je jak o těchto bezpečnostních požadavcích, tak i o vhodných nástrojích a metodách.

Největší hrozbu systému, jako vždy, představují koncoví uživatelé. Řadu problémů lze eliminovat technickými opatření (např. jednotnou správou programového vybavení klientských stanic), ale mnohá z nich nejsou technicky, finančně nebo psychologicky únosná. I zde je nutno uživatele vychovávat k bezpečnosti, a to se zdaleka nemůže týkat jenom hesel nalepených na klávesnici. Nejen administrátor musí totiž myslet trvale například na to, že manažer přistupující z domova k podnikové síti přes dokonalou IPSEC VPN či přes SSH představuje pro síť zásadní nebezpečí, pokud si do jeho stroje před tím stáhnul synek z Internetu trojského koně.

Obecně je tedy nutno trvale vzdělávat všechny aktéry a uživatele ICT s tím, že žádný z nich nesmí v bezpečnosti spatřovat překážku, ale musí ji zahrnout do příslušných procesů stejně jako třeba funkční požadavky systému.

4 Co nezanedbat

Na závěr ještě několik útržkovitých doporučení o tom, na co by se při budování bezpečnosti nemělo zapomenout:

- Systém řízení bezpečnosti ISMS podle BS 7799 s důrazem na členění odpovědností, na procesní vnímání bezpečnostni, na audity apod., by měl být začátkem komplexního řešení;
- Celopodnikový centralizovaný systém pro identifikaci a kryptograficky silnou autentizaci uživatelů (a pokud možno i autorizační systém) by měl zajistit ekonomicky efektivní a přitom dostatečně silný systém pro řízení přístupu k aktivům organizace;
- Perimeter defense + obrana do hloubky (osobní firewally a AV systémy) s jednotnou centralizovanou politikou pro řízení všech bezpečnostních opatření;
- Procedury a nástroje pro incident response, analýza výstupů FW a AV systémů, organizační opatření, BCP;
- Centrální politiku a mechanismy pro aktualizaci programového vybavení, pro updaty FW databází, AV signatur, IDS signatur apod.

A poznámku úplně na závěr:

BEZPEČNOST JE TRVALÝ PROCES!

Identifikační systémy v praxi v ČR po roce 2000

Jan Abel

E-mail: abel@ans.cz

Abstrakt

Se současným prudkým rozvojem podnikových informačních systémů a datových sítí stále zřetelněji vystupuje do popředí problematika jejich ochrany. A to i ve smyslu inteligentního řízení přístupu k těmto systémům. Známá prastará kombinace „jméno + heslo“, která se dnes v celém Internetu používá k ověřování identity uživatele, je pro použití v moderním vnitropodnikovém informačním systému naprosto nedostačující. Její jasnou slabinou je přílišná jednoduchost jejího uhodnutí a následná (praxí ověřená) vysoká četnost jejího vyzrazení. Moderní informační systém potřebuje něco více: důvěryhodnou komponentu, která by s maximální mírou zabezpečila jednoznačnost a spolehlivost při autorizaci uživatele ke vstupu do systému resp. ke zpřístupnění jeho jednotlivých funkcí. Článek ukazuje, jak lze problematiku důvěryhodných identifikačních systémů řešit v současné praxi.

Klíčová slova: PKI, čipová karta, Active direktory, Certifikační autorita, kryptografický čip

Moderní identifikační systémy můžeme dělit z různých pohledů (určení, zranitelnost, spolehlivost, cena, atd.), které víceméně kopírují dělení z pohledu nákladů na jejich pořízení:

- systémy **s jednoduchou identifikací** uživatele,
- systémy **s důvěrohodnou identifikací** uživatele,
- systémy **s biometrickou identifikací** uživatele.

První skupinu představují systémy s identifikací uživatele pomocí známé dvojice „*jméno + heslo*“. Jejich slabina je zřejmá: příliš vysoké riziko vyzrazení a následného zneužití příslušného účtu uživatele.

Druhou skupinu tvoří systémy, které se snaží užití dvojice „*jméno + heslo*“ nahradit něčím lepším (účinnějším, bezpečnějším), co by ztížilo možnost zneužití, ale zároveň zbytečně moc nezatěžovalo pravého uživatele. Obvykle jde o spojení s vlastnictvím nějakého předmětu resp. prostředku (obvykle elektronicky činného), který je aktivní na základě zadání nějakého vstupního kódu. Což v praxi znamená přechod ke kombinaci: „*vlastnit předmět + znát heslo*“.

Třetí skupinu pak představují systémy, které dokáží rozpoznat uživatele na základě jeho biometrických znaků (otisk palce, obraz sítnice, stavba obličeje, apod.). Tyto systémy jsou však pro svou vysokou pořizovací cenu v praxi využívány jen pro speciální účely.

1 Zlatá střední cesta: kombinovaná čipová karta

Opomineme-li fakt, že každý systém lze nějakým způsobem obelstít (je jen otázka, kolik času a peněz tomu můžeme věnovat), a odmítneme-li identifikační systémy na bázi biometrických prostředků (jako neúměrně drahé našim potřebám), vypadá takovýto kombinovaný systém identifikace, založený na nutném vlastnictví nějakého předmětu a znalosti autentizačního kódu (hesla), jakožto „zlatá střední cesta“ docela nadějně. Jeho spolehlivost je potom dána zejména ztížením možnosti daný předmět půjčovat – a to všemi dostupnými prostředky. A jako téměř ideální prostředek pro tyto účely se v současné době ukazuje **kombinovaná čipová karta**.

Proč? Důvody jsou nasnadě – a to jak technologické, tak i psychologické. Technologicky se totiž jedná o dnes již poměrně dobře zvládnutou problematiku, která nemá nijak extrémně vysoké finanční nároky na realizaci. Ale **hlavní důvody pro její užití jsou v oblasti psychologické**: dnešní člověk je i v soukromém životě navyklý používat kartičky všeho druhu (vizitky, platební karty, identifikační visačky, magnetické karty, kryptografické karty, atd.) tak často, že mu to připadá normální, a dalšímu takovémuto systému se tedy ze zásady nebrání.

Tohoto psychologického faktoru můžeme s úspěchem využít. Moderní identifikační systém založený na kombinované identifikační kartě se přitom může sestávat z několika na sobě nezávislých subsystémů, které se

ale dovedou vzájemně podporovat. Stejná identifikační karta pak může sloužit k řízení přístupu do objektů resp. prostor s omezeným pohybem osob, evidovat výdej obědů, otevírat závoru na služebním parkovišti, řídit přístup k počítačovému systému, šifrovat a dešifrovat citlivá data, či elektronicky podepisovat daňová přiznání.

Že to není triviální problém, je asi každému hned jasné. Není to zas ale žádná velká teorie z daleké budoucnosti. Takovéto systémy dnes již existují. A existují nejen někde za oceánem (v laboratořích velkých počítačových firem), ale postupně vznikají i u nás doma, v Česku.

2 Čtyři systémy na jednom médiu

Budujeme-li identifikační systém, založený na identifikaci uživatele pomocí metody „*vlastnit předmět + znát jeho aktivační kód*“, musíme mít na zřeteli, že jedním z nejdůležitějších parametrů celého systému je jeho schopnost aktivně zabraňovat uživatelům v půjčování předmětu, na který je vázána identifikace uživatele, dalším osobám resp. v jeho odkládání na nechráněných místech. Využití kombinované čipové karty přitom pro tyto funkce se samo nabízí.

Kombinovaná čipová karta v sobě může zahrnovat až 4 na sobě nezávislá média pro 4 různé subsystémy, jejichž činnost se navzájem podporuje. Jsou to:

potisk – potiskem lícové strany zajistíme jednoduše základní funkci identifikační karty, kterou je **vizuální identifikace**. Její součástí obvykle bývá jméno organizace, jméno a osobní číslo uživatele, podobenka (fotografie) uživatele, apod.,

magnetický proužek – toto dnes spíše historické médium se v současnosti užívá spíše z důvodu kompatibility se staršími identifikačními systémy, především pro řízení přístupu osob do nekritických prostor resp. aplikací,

bezkontaktní kryptografický čip – moderní médium, které se využívá především pro systémy kontrolující (řídící) přístup do prostor s omezeným resp. monitorovaným pohybem osob,

kontaktní kryptografický čip – moderní médium pro ukládání elektronických certifikátů.

A v tomto momentě nastupuje výše zmiňovaná psychologie: zkombinujeme-li technologii řízení přístupu do budovy pomocí bezkontaktního čipu (případně magnetického proužku) a využijeme-li kontaktní čip pro uložení elektronických certifikátů, můžeme ho s výhodou využít pro řízení přístupu do počítačového systému. Protože uživatel, který má kartu vloženou do čtečky v počítači, nemůže odejít z budovy, jelikož nemá čím si otevřít dveře. A když se pro kartu vrátí a z počítače jí vytáhne, operační systém počítače danou stanici uzamkne, takže nemůže dojít k případu, že by s aplikací běžící v dané chvíli na uživatelské stanici pracoval někdo jiný. Zkombinujeme-li tuto situaci navíc s dnes obvyklým nařízením, že identifikační karta každého zaměstnance musí být při pohybu osob po budovách nošena stále viditelně připnuta k oděvu a že v případě ztráty karty je tuto ztrátu uživatel povinen neprodleně ohlásit např. na příslušný bezpečnostní referát, těžko se pak uživatel může vymlouvat na to, že někdo rozlousknul jeho heslo a pod jeho účtem vymazal data z databáze (tj. že on za nic rozhodně nemůže, že to musel udělat někdo jiný, kdo se za něj vydával).

A právě v tomto psychologickém aspektu je zakleto celé kouzlo moderního identifikačního systému!

Kombinovaná čipová karta proto představuje ideální identifikační prostředek, který zaměstnanec používá pro všechny důležité úkony. Vstupem do budov, místností a dalších prostor počínaje, přes automatizovanou evidenci docházky, placení obědů, fasování materiálu, až po řízení přístupu k informačním systémům s možností využití technologií šifrování a digitálních podpisů příslušných dokumentů.

3 Technické řešení skoro zadarmo

Že realizace solidního identifikačního systému nemusí představovat milionové finanční náklady, si ukažme např. na řešení postaveném nad platformou Windows 2000 + Exchange 2000, která je mezi podnikovými systémy hojně zastoupena.

Základem takového řešení je Windows 2000 doména s funkční Active Directory, která z principu slouží pro ukládání veškerých informací, sloužících k řízení přístupu do počítačových systémů do této domény připojených. V našem řešení s výhodou využijeme její hlavní přednost – propracovanou a spolehlivou funkčnost synchronizace jednotlivých kopií

distribuované databáze účtů, podporovanou přímo na úrovni OS (kontrukce Active Directory nám zaručuje, že budeme mít k dispozici veškerá data ohledně uživatelských účtů a přístupových práv ve všech lokalitách neustále synchronní).

Do Active Directory proto uložíme organizační strukturu podniku a účty jednotlivých uživatelů propojíme s úložištěm elektronických certifikátů. Od verze Windows 2000 SP3 systém funkci přihlašování uživatelů pomocí elektronických certifikátů přímo podporuje (od verze SP4 ji pak podporuje bez viditelných problémů). Stejně tak i SW balík Office 2000 (a jeho novější verze) přímo podporuje funkce pro šifrování a elektronický podpis dokumentů pomocí elektronických certifikátů. Rozhodneme-li se pro čipovou kartu (jakožto média pro ukládání elektronických certifikátů), potřebujeme ke každé klientské stanici vlastně dokoupit jen čtečku čipových karet.

Pro větší implementace (100 a více uživatelů) se však doporučuje ještě jedno praktické rozšíření: zřízení samostatného pracoviště, tzv. Registrační autority, jehož úkolem je centralizované vydávání el. certifikátů pro celou organizaci. Technicky je možné takovéto pracoviště zřídit jako pobočku registrované (obecně uznávané) certifikační autority. V takovém případě je systém bez problémů vydávat i tzv. kvalifikové certifikáty – tj. certifikáty, kterézto jediné je možné podle zákona o el. podpisu užívat při komunikaci se státní správou.

V této konfiguraci je pak možné (s využitím poštovního systému Exchange 2000) úplně zautomatizovat periodické přebírání seznamu zneplatněných certifikátů, který je příslušná certifikační autorita povinna udržovat a pravidelně (denně) aktualizovat. Uvedené řešení má další praktickou výhodu: rozhodneme-li se později sdílet důvěru v námi vydané certifikáty s jinými subjekty, nemusíme se nijak starat o agendu spojenou se vzájemným předáváním seznamů zneplatněných certifikátů, protože o tuto funkcionalitu se ze zákona stará příslušná nadřazená certifikační autorita, a tak jednoduše veškeré dotazy externích subjektů na platnost námi vydaných certifikátů směřujeme na ni.

Výše popisované řešení asi nejvíce ocení programátoři, vyvíjející aplikační SW pro Windows. Microsoft totiž problematiku užívání el. certifikátů k autorizaci veřejně podporuje a tak pro programátory připravil účinné nástroje přímo pro potřeby technologie PKI (Public Key Infrastructure) – a to přímo ve formě rozšíření funkčnosti standardního Windows API. Několik málo DLL knihoven, souhrnně označovaných jako

Crypto-API, obsahuje vše potřebné pro jednoduché psaní aplikací, které tuto moderní technologii využívat chtějí resp. potřebují.

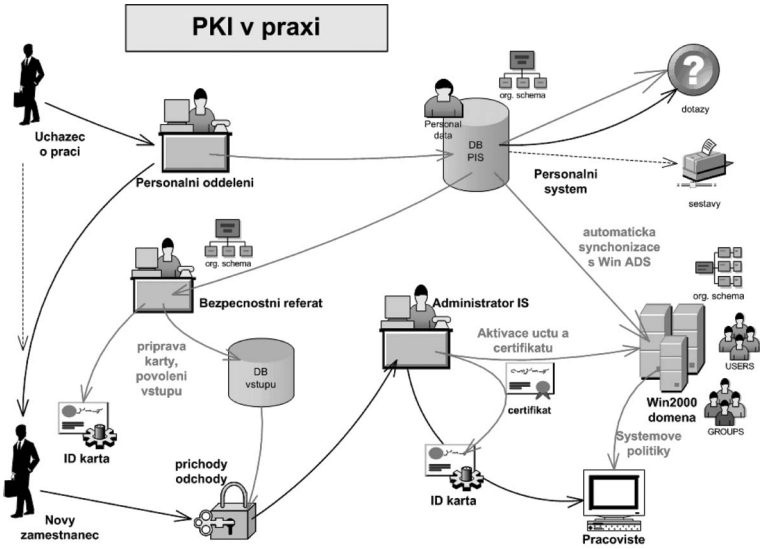
4 Technické řešení ale není všechno

Naším úkolem však je vybudovat dlouhodobě funkční identifikační systém, který bude všechny své funkce plnit spolehlivě a stabilně – tj. zejména bude schopen rychle a účinně se bránit proti úmyslnému zneužití identity vybraného uživatele. Máme na mysli především případ úmyslného zcizení identifikační karty uživateli s cílem proniknutí do budovy resp. informačního systému.

Proto je nutné vlastní technické řešení na poli informačního systému doplnit řadou podpůrných funkcí, jak technického, tak netechnického charakteru. Jedná se zejména o problémy:

- kdo a jak bude čipové karty zaměstnancům vydávat,
- kdo a jak bude aktivovat jejich jednotlivé subsystémy,
- jak bude zajištěno blokování jednotlivých subsystémů, v případě ztráty nebo zcizení karty,
- za jakých podmínek bude resp. nebude požadována úprava aplikačního SW, který by běl být v budoucnu na tuto technologii také propojen,
- jak budou zajištěny náhradní mechanismy (např. potřeba nouzově povolit přístup do informačního systému bez užití karty) pro uživatele, který kartu ztratil (nebo mu během dovolené vypršela její platnost, apod.) a novou kartu ještě vystavenu nemá,
- jak se budou uchovávat a skartovat dokumenty, nutně obsahující osobní údaje jednotlivých zaměstnanců(uživatelů),
- atd. atd.

Teprve potom budete si moci říci, že máte v provozu moderní identifikační sytém integrovaný s podnikovým informačním systémem s funkčností například dle následujícího obrázku.


PKI v praxi
Uchazec o praci
Personalni oddeleni
org. schema
Personal data
DB PIS
Personalni system
dotazy
sestavy
automaticka synchonizace s Win ADS
org. schema
Bezpecnostni referat
org. schema
Administrator IS
Aktivace uctu a certifikatu
priprava karty, povoleni vstupu
DB vstupu
certifikat
USERS
Win2000 domena
GROUPS
Systemove politiky
ID karta
prichody odchody
ID karta
Novy zamestnanec
Pracoviste

Útoky na klienta

Juraj Bednár

E-mail: juraj@bednar.sk

Úvod

V tejto prednáške chcem priblížit casto opomínanú problematiku bezpecnosti klienta. Aj napriek súcasným trendom presúvania dôležitej funkcnosti informacných systémov do serverov a mainframov, klientské stanice pracovníkov a obzvlášt systémových administrátorov sú dôležitými prvkami pri posudzovaní bezpecnosti informacného systému. Vzhladom na to, že klientské stanice sú ovládané väcšinou používatelmi s nízkymi znalostami informacných technológií a obzvlášt bezpecnosti, mnohokrát je najjednoduchším spôsobom prieniku útok na klientskú stanicu.

Riadenie vnemov

Medzi najbežnejšími bezpecnostnými rizikami bol oddávna ludský faktor. Voci nemu je možné robit útoky s prvkami riadenia vnemov (castokrát trošku nesprávne nazývame túto techniku sociálne inžinierstvo). Riadenie vnemov v kombinácii s technologickým útokom sa ukazuje ako jedna z najúcinnejších metód útokov vôbec.

Príkladom najbežnejšieho útoku s použitím riadenia vnemov je posielanie programov. Predstavte si, že úctovnícke do firmy príde CD s potlacou ich úctovného programu a s vysvetlením, že ide o novú verziu zohladnujúcu aktuálne právne normy. Nielenže je dost pravdepodobné, že program na takomto CD úctovnícka spustí, je dokonca velmi pravdepodobné, že by ho spustil aj systémový administrátor, ktorý sa venuje inštalácii softvéru. Využitý je fakt – ktorý nám mimochodom poslúži aj neskôr – že mnohé softvérové distribucné kanály sú netransparentné a castokrát nezabezpecené. V prípade, že sa nám na klientskej stanici podarí spustit kód, ktorý chceme, máme vyhraté. Menšou obmenou je posielanie trójskych koní elektronickou poštou, táto metóda sa však vdaka antivírovej ochrane a blokovaniu spustitelných programov na strane servera využíva coraz menej.

Samozrejme, nesmieme zabúdat na „informacný šum“, ktorý môže vzniknút. Útocníkovi môže pomôct už len samotný fakt, že dokáže používatelovi vnútit nesprávnu informáciu. To sa môže diat pomocou mailu, webu alebo inými cestami.

Man in the middle útok v spojení s riadením vnemov

V prípade, že sa útocník vie dostat medzi klienta a server a modifikovat dáta, ktoré sú po ceste, je možné uskutocnit viacero útokov.

Azda najmenej používaným – no pravdepodobne najmenej riešitelným problémom je modifikácia informácií. Pomocou šikovne naprogramovaného filtra je možné modifikovat celé stránky. Modifikácia môže prebiehat takmer úplne transparentne. Môže sa stat, že klient pri návšteve stránky, ktorú pravidelne navštevuje a dôveruje jej, dostane podsunuté nesprávne informácie, ktoré ho môžu prinútit k rôznym reakciám (napríklad konkurenciou podhodená informácia vrámci odboru môže prinútit používatela k nesprávnemu pracovnému rozhodnutiu). V súcasnej dobe zamestnanci castokrát väcšinu dôležitých informácií získavajú cez web.

Infikácia stahovaných programov

Pri dobre naprogramovanom filtri nie je problém automaticky infikovat stahované spustitelné súbory kódom, ktorý umožní útocníkovi prístup na klientskú stanicu. Infekcia môže byt robená prúdovo (teda za chodu, nie je potrebné dotiahnut celý spustitelný súbor). Na pôvodný súbor je vhodné použit kompresiu s doplnením na pôvodnú velkost súboru. Používatel si tak myslí, že stahuje program z dôveryhodného zdroja, akurát sa k nemu nieco pribalí. Osobne si myslím, že aj ked vzrastá pocet softvérových firiem, ktoré používajú digitálny podpis, spolahnút sa nan nedá. Dokonca ani mnohé antivírové firmy nepoužívajú digitálny podpis balíkov, ktoré distribuujú. O užitocných freewarových aplikáciách to platí dvojnásobne – a takmer každý z nás aspon z casu na cas nieco také inštaluje.

Taktiež je možné za chodu modifikovat javový kód, spúštaný v appletoch (príkladom môže byt projekt Mobile Code Security Through Java Byte-Code Modification:
http://theory.stanford.edu/∼vganesh/project.html).
Techniky infikácie ELF nájdeme zase v casopise Phrack:
http://www.phrack.org/show.php?p=61&a=8

Public Key Infrastructure

Na prekonanie týchto dvoch problémov vznikla PKI (Public Key Infrastructure). Snaží sa sklbit technológie asymetrického šifrovania a digitálneho podpisu do infraštruktúry dôvery, ktorá by mala zarucit bezpecný prenos dát (ci už informacných alebo programových). Aj napriek tomu väcšina poskytovatelov tento prístup neposkytuje, prípadne ho poskytuje so "samopodpísaným" certifikátom, ktorý je možný jednoducho falšovat (napr. programy zo sady dsniff dokážu robit man in the middle na https spojenie).

V prípade, že sa táto infraštruktúra používa, je možné spravit niekolko útokov. Najjednoduchší útok je vygenerovat nový klúc podpísaný nesprávnou certifikacnou autoritou a dúfat, že používatel je natolko neuvedomelý, že varovanie ignoruje. Pokrocilejším spôsobom je útok na certifikacnú autoritu. Mnohí si myslia, že je to nereálne a že certifikacné autority patria medzi najbezpecnejšie organizácie. V roku 2000 sa mi podarilo použit niektoré techniky útokov na klienta (v tomto prípade na cloveka rozhodujúceho o udelenie certifikátu) a bol mi vydaný platný, podpísaný certifikát, ale viac-menej neprávom (bol to pokus, na ktorý som samozrejme mal oprávnenie od majitelov serverov, ktorých certifikát som chcel získat). Presný postup bol publikovaný v casopise 2600.

Prvou certifikacnou autoritou, ktorú sa mi podarilo presvedcit, aby mi neprávom vydala certifikát bola spolocnost GeoTrust. Na overenie pravosti potrebujú Business License (v našom prípade výpis z obchodného registra). Ak odhliadneme od faktu, že u nás je možné získat kohokolvek výpis z OR (stací vyplnit kolkovanú žiadost) a že výpis je platný len s obcianskym preukazom konatela alebo osoby oprávnenej zastupovat danú organizáciu (co samozrejme v GeoTruste nevedeli), skúsil som iný postup. Požiadal som o testovací certifikát (ktorý sa od "ostrého" odlišuje len tým, že má dobu platnosti niekolko dní, namiesto rocnej platnosti, je však plne korektný) a zároven som povedal majitelovi servera, nech si u danej certifikacnej autority požiada o certifikát. On poslal všetky potrebné papiere a mne udelili certifikát. Možnost, že by sa dvaja rôzni ludia zaujímali o ten istý certifikát im pravdepodobne prišila pomerne nereálna.

Druhá certifikacná autorita (ktorá je naštastie nainštalovaná štandardne len v Microsoft Internet Exploreri a nie v Mozille) certifikát vydávala na základe zaslania e-mailu na administratívny kontakt domény.

Fakt, že v európe sú štandardne kontaktné záznamy neautentifikované im asi tiež nepripadal velmi zaujímavý. Stacilo jeden e-mail do ripe, administratívny kontakt bol zmenený a mne prišiel e-mail s linkou, na ktorú som klikol. O pár minút mi bol vydaný certifikát. Druhá dôležitá vec je, že v ripe sa nachádzajú aj technické kontakty a tento kontakt je možné použit tiež – teda provider by aj v prípade, že doména používa autentifikované záznamy, bol schopný získat certifikát.

Nevýhodou tohto prístupu je, že je možné robit man in the middle útok len na konkrétne domény. Druhou možnostou je využit predchádzajúci útok a po spustení kódu nainštalovat certifikát fiktívnej certifikacnej autority priamo do prehliadaca a takto robit man in the middle útok na všetky zabezpecené kanály (vrátane možnosti modifikácie prechádzajúcich dát).

Treba si uvedomit, že šifrovanie nie je len jedno tlacidlo s nakreslenou zámkou, ale proces, ktorému je potrebné aspon zcasti rozumiet. Toto chciet po bežných používateloch sa však ukazuje ako nereálne.

Techniky man in the middle

Technická realizácia man in the middle útokov závisí hlavne od prenosového média. V ethernete a pri technológii wifi je možné použit arp spoofing na presmerovanie klienta (napr. Pomocou programu arpspoof z balíka dsniff). Samotnú modifikáciu HTTP a HTTPS dát je možné robit pomocou špecializovaného softvéru (napr. ettercap) alebo pomocou presmerovania na filtrujúci proxy server. Takto je možné robit aj „interaktívnu“ modifikáciu, pri ktorej sa stránka zobrazí najprv útocníkovi, ten pomocou skratiek stránku rýchlo zmodifikuje a pošle klientovi.

Od prenosového média nezávislou technikou je napr. DNS spoofing.

Backdoory a trójske kone

V prípade zmeny kódu si musíme položit otázku, ako najlepšie ovládnut klientskú stanicu. Je viacero programov, ktoré sa týmto zaoberajú. Azda najznámejším je Back Orifice (http://bo2k.sf.net/). Jeho nevýhodou je, že je pomerne známy a teda ho rozoznáva väcšina antivírových programov. Nie je však problém spravit modifikáciu, ktorá znemožní detekciu (kedže k Bo2K sú prístupné zdrojové kódy). Druhou možnostou je použitie metasploit payloadov (http://www.metasploit.org/). K použitiu týchto backdoorov je však potrebná lepšia znalost programátorských techník.

Castým problémom, na ktoré trójske kone v súcasnosti narážajú, sú osobné firewally pod OS Windows. To, že neobsahujú žiadnu rozumnú ochranu (v spolupráci s dost slabým bezpecnostným modelom operacného systému Windows) sa môžeme dozvediet na stránkach casopisu Phrack v clánku Using Process Infection to Bypass Windows Software Firewalls (http://www.phrack.org/show.php?p=62&a=13).

Pomocou backdoorov a trójskych koní je možné úplne ovládnut klientskú stanicu a získat prístupy na ostatné miesta v informacnom systéme, prípadne použít danú stanicu ako vstupnú bránu na iniciovanie dalších útokov.

Zdanlivá ochrana klientov

Castokrát sa zdá, že provider ochranuje zákazníkov pred útokmi tým, že im pridelí adresu z privátneho adresného priestoru a teda nie je možné zvonku iniciovat spojenie. Takto sa casto stáva, že používatelia majú slabo nastavené bezpecnostné pravidlá. Typickým a castým príkladom sú GPRS provideri: stací sa pripojit cez poskytovatela, tiež získat adresu z privátneho adresného priestoru a poobzerat sa po ostatných pripojených používateloch. Takto je možné realizovat aj predtým nerealizovatelný útok na klienta, ktorý bol zdanlivo chránený. Využíva sa tu casto opomínaný fakt, že siet je málokedy chránená aj proti útokom znútra.

Záver

Klientské stanice patria k miestam s najväcšími reálnymi bezpecnostnými rizikami. Nevzdelanost používatelov a mnohé nedoriešené problémy zvyšujú riziko úspešného útoku.

Literatúra

[1] Phrack 62: Bypassing Windows Personal FW‘s.

[2] Phrack 62: Advances in Windows Shellcode.

[3] Phrack 62: Kernel Mode Backdoor for NT.

[4] Phrack 61: The Cerberus ELF interface

Hardware accelerated encrypted connection

Jiří Novotný, Milan Sova

E-mail: novotny@ics.muni.cz, sova@cesnet.cz

Abstrakt

Among many characteristics demanded of today's Internet, two seem to stay in strong opposition: high data throughput and their security. While the latter is achieved by strong encryption demanding high computing power, the former suffer from any delay. We presume that both could be fulfilled by introducing hardware crypto accelerators.

In this article, we assume a network interface card capable of performing symmetric encryption/decryption. We will consider integration of such a device to existing traffic encryption scenarios.

1 Cryptography in Internet traffic protection

Cryptography may be (and, actually, is) used to secure data in many different places. We will provide basic classification of cryptography usage for securing Internet traffic with respect to possible employment of hardware encryption accelerators.

1.1 Link level encryption

Link level encryption provides a secured channel between two network nodes by applying encryption at the link level generally via an external crypto device. The nodes' network stacks are minimally influenced by this configuration since all of the security functions are displaced to the device. The link level encryption even enables the communication parties to use arbitrary protocol (i.e. they are not restricted to use

e.g. TCP/IP). However, this separation of the operating system and the crypto device as well as lack of standardized protocols for channel creation and management may in some situations be viewed as aádisadvantage.

On the other hand, link level encryption provides strong security by effectively creating isolated connection layered over public network guaranteeing not only confidentiality and authenticity of the transfered data but also minimizing possible traffic analysis.

Link level encryption is usually used for creating static peer-to-peer secured channels or virtual private networks connecting two cooperating sites. In any case, both peers should be known to each other in advance and should cooperate in setting the connection up generally using some out-of-band means of communicating the channel security parameters. This provides for strong security but at the same time restrains from creating ad-hoc connections with the flexibility encountered in common Internet usage.

A link level encryption accelerator does not require any specific cooperation with operating system of the host it provides encryption/decryption to. Its implementation is therefor relatively simple compared with encryption devices operating at higher levels.

1.2 IP level encryption – IPsec

Security deficiencies inherent in IP since the beginning are rectified by the suite of IP Security Protocols (IPsec) [RFC2401], [RFC2402], [RFC2406].

IPsec defines two new optional headers to IP packets, Authentication Header (AH) and Encapsulation Security Payload (ESP), providing authenticity and data integrity and confidentiality services to individual connections. Each of these headers carries a Security Parameters Index (SPI) – a numerical value which, in combination with packets' IP Destination Address and a security protocol (AH or ESP), uniquely identifies a security association (SA) – a simplex logical "connection" providing specific security processing.

Authentication header provides connectionless integrity and data origin authentication for IP datagrams. AH contains a cryptographic checksum of the datagram contents. The checksum is computed over the immutable fields of the datagram's IP header, the AH header itself (with the Authentication Data field to zero), and the upper level protocol data.

ESP may provide, among other security services, confidentiality (it may be used to provide data origin authentication, connectionless integrity, an anti-replay services, and limited traffic flow confidentiality as well). When confidentiality is selected, the ESP header contains in its Payload Data field the datagram's payload encrypted using algorithm and parameters defined in the current SA.

IPsec protocols may be applied in one of two modes: transport mode and tunnel mode. In transport mode, the IPsec headers are included into the original IP datagram after the IP header. The source and destination of the datagram listed in the IP header are not protected by ESP and may be vulnerable to traffic analysis. On the other hand, the communication overhead is smaller than for tunnel mode.

In tunnel mode, the technique of packet encapsulation known as IP-in-IP tunneling is used. The whole original datagram is processed by ESP and/or AH and the outcome of the operation is used as data payload of a new datagram compound of a new IP header, IPsec headers, and the payload. This technique enables addressing the new datagram to IPsec gateway providing security services for a group of hosts (usually a local network) and creating a Virtual Private Network in this way.

The IPsec traffic processing is driven by two databases implemented for every IPsec-enabled interface:

Security Policy Database (SPD) SPD specifies which services are to be offered to IP packets and in what fashion. It is consulted during the procession of all traffic (both incoming and outgoing). For any datagram, the consultation process leads to one of three possible results:

discard the packet is silently discarded,

bypass IPsec the packet is allowed to continue without any IPsec processing,

allow IPsec the traffic will be IPsec protected. In this case the security services protocols, algorithms etc. are provided by the SPD.

Security Association Database (SAD) Each entry in SAD defines parameters associated with one SA. In case of outbound traffic,

SAD entries are accessed by pointers from entries in SPD. For inbound traffic, the SAD entry is looked up via an index constructed from destination IP address, IPsec protocol type and SPI.

SAD entries contain information used for IPsec processing, namely:

- Sequence Number Counter,
- Sequence Counter Overflow,
- Anti-Replay Window,
- AH Authentication algorithm, keys, etc.,
- ESP Encryption algorithm, keys, IV mode, IV, etc.,
- ESP authentication algorithm, keys, etc.,
- Lifetime of this Security Association,
- IPsec protocol mode: tunnel, transport or wildcard,
- Path MTU.

1.2.1 IPsec key management

Both AH and ESP rely on existence of Security Associations but neither of them deals with creation of SAs. RFC 2409 [RFC2409]defines the Internet Key Exchange (IKE) protocol responsible for negotiating SA and providing authenticated key material for them. IKE operates in two phases: Phase 1 establishes an Internet Security Association and Key Management Protocol (ISAKMP) [RFC2408] SA – a secure channel through which the IPsec SA negotiation takes place. Phase 2 establishes the actual IPsec SAs. Phaseá1 provides strong mutual authentication of parties (usually involving asymmetric cryptography) and encryption for the ISAKMP SA as well as a shared secret involved in establishing keying material for the Phase 2.

1.2.2 Hardware acceleration of IPsec

IPsec operates at the IP level, therefor the natural place for its implementation is in the native IP stack. Several software implementations of IPsec for open-source operating systems are available (FreeS/WAN http://www.freeswan.org/, KAME http://www.kame.net/, OpenBSD http://www.openbsd.org/, . . .) and can be modified to employ a hardware crypto device.

1.3 Application level encryption – SSL/TLS

Secure Socket Layer (SSL) ([SSL3], [SSL2]) is an application layer protocol originally developed by Netscape Communications to protect HTTP communication. SSL3 was further developed by IETF and the outcome became standardized as Transport Layer Security (TLS) ([RFC2246]). TLS is nowadays used to secure traffic for various application protocols like HTTP, IMAP, POP3, LDAP, SMTP, and others.

TLS provides authenticity, confidentiality, and key exchange. The TLS protocol itself is layered, with four defined protocols:

The handshake protocol optionally authenticates peer's identity and negotiates security parameters (keys, initialization vectors, and MAC secrets) for the session.

The alert protocol is used to communicate failures, errors, and notification between parties.

The change cipher specification protocol is used to signal transition in ciphering strategies.

The application data protocol carries the fragmented, compressed and encrypted application data.

The Record Layer (the "lower" layer of the protocol serving as a transport for the protocols above) receives data from higher layers and prepares them for transmission. The data are processed in following steps:

Fragmentation By coalescing several client messages into one record or fragmenting a client message into several records, the record layer ensures that the record length will not exceed 2^14 bytes.

Record compression/decompression The record is compressed using compression algorithm defined in the current session state.

Record payload protection The MAC of the compressed record is computed using algorithm and key material defined in the current session state.

Record encryption The data including the MAC value are encrypted using algorithm and key material defined in the current session state.

1.3.1 Hardware acceleration of SSL/TLS

Implementing hardware crypto acceleration for SSL/TLS faces a serious problem. SSL/TLS being an application protocol is provided by several applications/tools/libraries which may run concurrently on one host. A typical Linux workstation is usually running a Mozilla http://www.mozilla.org/ – derived web browser using its own SSL library, some applications using OpenSSL http://www.openssl.org/ toolkit and maybe some other using GnuTLS http://www.gnu.org/software/gnutls/. Mozilla's Netscape Security Services (NSS) as well as OpenSSL enable using hardware devices for performing crypto operations. Although their APIs differ, none of the libraries assume cooperation with encryption enabled network interface device.

2 Integration of encryption capable network interface card into operating system

Cryptographic operation are computationally demanding, yet the need for secure transferring of large data is constantly growing. One of possible ways to reduce the host CPU load and to increase data throughput is to move the data encryption/decryption processing to crypto enabled network card. For the purpose of this article, we consider integrating a network interface card capable of performing symmetric cryptographic operations into existing traffic encryption schemas implemented in software. We won't discuss the link level encryption for its lack of scalability.

Both protocols considered here, i.e. IPsec and TLS, need asymmetric cryptography for authentication and key management. Asymmetric crypto functionality is much more demanding of CPU compared to symmetric encryption/decryption, however authentication and key management operate on small data units. The ratio between asymmetric and symmetric crypto operations needed to transfer data through a secured channel depends on used protocol, its parameters and on the application. A HTTPS server providing HTML pages to a large amount of clients is expected to spend relatively more time performing asymmetric crypto operations than an IPsec gateway connecting local clients to a remote site. For the case of simplicity we further assume that asymmetric crypto operations are performed in software or in some other dedicated

hardware and that only symmetric encryption/decryption is performed by the network card.

Combination of encryption/decryption accelerator with network interface aims at minimizing the number of data transfers through system bus. Since existing software implementation of the above mentioned protocols is assumed to provide connection management, a communication of key material between the software and the card must be decided. Minimization of system bus usage requires storing the key material for individual sessions on the network card. This implies modification of the software and providing an API for accessing the session data. The natural candidate for the API is the PF_KEY Key Management API [RFC2367] used in IPsec implementations by key management application (usually an IKE daemon) to communicate with an operating system's key management internals (Security Association Database). PF_KEY provides for creating, modifying, inquiring, and deleting SAs. In the case of IPsec, the software modification required represent replacing kernel SAD interface with a new one proxying PF_KEY communication to the accelerator.

For TLS, the software session management implementation should be replaced in a similar fashion using PF_KEY with some extensions for the card API. Although developing of a new, TLS specific, card API may be possible, PF_KEY may provide the needed functionality for both protocols.

Software passing data to the card for transmission must be modified to skip the crypto processing and to instruct the accelerator what crypto operations it has to perform. Prepending a specific header (or headers) indicating the required crypto operations by pointing to the corresponding SA (or session, for TLS) is a very natural way to achieve the task. On receiving the data the card strips these headers, looks up the relevant status SA (session), performs the crypto operations, and send the newly created IP datagram to the peer. On receiving a datagram from the network the card looks up the SA (or TLS session) using information from the datagram, performs the required crypto operations, and passes the processed packet to operating system prepended with headers indicating which SAs (or TLS sessions) were applied so that policy database can be consulted in software.

3 Hardware acceleration in FPGA

3.1 Why the hardware accelerated encryption

The need for hardware acceleration for crypto operations results from the disproportion between ever increasing data transmission speeds and the power of CPUs. While today's high speed networks operate at 1–10 Gb/s, the fastest reported C/C++ implementation of AES achieve the speed 0.4 Mb/MHz which means that 3.2 GHz Pentium can encrypt at most 1.28 Gb/s of data (and the processor is fully occupied by the encryption). AES implemented in FPGA can operate at 129.6 Mb/MHz [SLC02]. This is 12.96 Gb/s for 100 MHz XILINK VIRTEX II family FPGA. Such a speed is enough even for 10 Gb/s network.

An improved AES implementation was reported to achieve 20 Gb/s encryption speed in older VIRTEX family [SMMS03]. Triple DES can achieve 21.3 Gb/s in VIRTEX II [RSQL03].

3.2 AES (Rinjdael algorithm)

The Rinjdael algorithm was selected by NIST as the Advanced Encryption Standard (AES) in October 2000 and approved as the Federal Information Processing Standard (FIPS) in the summer of 2001 [AES].

AES is a symmetric block cipher. The length of the Block and of the Cipher Key can be independently chosen equal to 128, 192 or 256 bits, but for official AES version, the only legal block length is 128 bits[SRD03]. More complete AES description including the mathematical background can be found in [BS91].

AES algorithm iterates Round transformation of the data. Intermediate result during the Round transformation is called the State (4 byte vector). The AES algorithm consist of four basic operations:

ByteSub: non-linear substitution based on Galois Field applied to each state byte.

ShiftRow: cyclically shifting the last three rows of the State by different offsets.

MixColumn: Galois Field polynomial multiplication is performed on each State column.

AddRoundKey: each round key byte is XORed with the corresponding State byte.

The single round is repeated 10 times. An additional round is applied in which the original key is added to the input data. The key entering

the cipher is expanded so that a different sub-key (round key) is created for each row of the algorithm. This round key generation is a process performing S-boxes, XORs and word rotation operations.

AES is perfectly suitable for hardware implementation, where the ByteSub is designed as a small 8×8 table, ShiftRow is done just by interconnection of State blocks, the MixColumn uses shifts and adds, and the last step uses XOR functions. All of that steps can be simply implemented in FPGA. The design of AES allows pipelining – technique which is very fast when used in hardware.

3.3 Implementation of hardware acceleration

The complexity of a hardware crypto accelerator depends on the network protocol used and is generally higher for higher levels of the network stack. We will discuss main ideas of the FPGA based crypto for some of them. We assume an "intelligent" NIC card based on a PCI card with Ethernet interface(s) and FPGA. The COMBO6 card [NFK02] with one of the add-on cards [NFK03a], [NFK03b], [COMBO] developed within the CESNET development activity "Programmable hardware" [PH] may be a good candidate for the hardware.

3.3.1 Link level encryption

Link level peer-to-peer encryption providing just the payload confidentiality using one shared key requires at least implementation of the encryption/decryption core and a simple state machine. In the simplest case the encryption key would be uploaded to the FPGA via its host computer driver. The implementation is straightforward, but using one shared key limits the employment of such a simple design.

Extending the hardware with a encryption/decryption keys database and generalizing the key uploading functionality enables communication with several peers using different keys. Adding a packet classification mechanism [AK04] can significantly decrease time expensive hardware – software communication.

3.3.2 IPsec

For IPsec, the design described in Section 3.3.1 has to be extended. A HMAC functionality must be added as well as IP packet manipula-

tion and editing routines. The edit engines may be implemented using nanoprocessors [NFA03]. The key database should be extended to provide full SAD functionality accessible via PF_KEY API provided by the host computer driver. Defragmentation routines and the corresponding buffers are necessary.

Incoming packets classification enables the policy-driven access control executed by the system.

3.3.3 Application level encryption

Of several widely used application level encryption protocols, SSL/TLS is the main candidate for hardware acceleration. In addition to the main building block used for IPsec, the TLS Record Layer may be implemented in hardware for efficiency. In that case a compression/decompression algorithms should be provided by the accelerator as well.

4 System description

Summarizing the discussion above we conclude that hardware accelerated crypto system may be built using following parts:
Software implementation modified to

- use SAD or session database on the accelerator using PF_KEY API
- replace the symmetric crypto functionality with tagging the data with appropriate SPI (session id)

Symmetric crypto enabled network interface card implementing:

- a strong symmetric cipher,
- a Hashed Message Authentication Code function,
- a Security Association Database accessible via PF_KEY API,
- IP fragmentation/defragmentation functionality.

References

[AES] *Specification for the Advanced Encryption Standard(AES)*, NIST, November 2001.

[AK04] Antoš D., Kořenek J.: *String Matching for IPv6 routers.* SOFSEM 2004, Matfyz Press, Praha 2004, pp. 205–210.

[BS91] Biham E., Shamir A.: *Differential cryptanalysis of DES-like cryptosystems.* Journal of Cryptology, Vol. 4, No. 1, 1991, pp. 3–72.

[COMBO] *COMBO-2XFP.* Liberouter, http://www.liberouter.org/card_combo2xfp.html

[Liberouter] *Liberouter project WWW page.* Liberouter, http://www.liberouter.org

[NFA03] Novotný J., Fučík O., Antoš D.: *Project of IPv6 Router with FPGA Hardware Accelerator.* Field-Programmable Logic and Applications (FPL), 2003, pp. 964–967.

[NFK02] Novotný J., Fuřík O., Kokotek R.: *Schematics and PCB of COMBO6 card.* Technical Report, 14/2002, CESNET, 2002.

[NFK03a] Novotný J., Fučík O., Kokotek R.: *Schematics of COMBO-4MTX card.* Technical Report, 13/2003, CESNET, 2003.

[NFK03b] Novotný J., Fučík O., Kokotek R.: *Schematics of COMBO-4SFP card.* Technical Report, 12/2003, CESNET, 2003.

[PH] *Programmable hardware.* CESNET, http://www.ces.net/project/details.html#a02

[RFC2246] Dierks T., Allen C.: *The TLS Protocol Version 1.0.* January 1999.

[RFC2367] McDonald D., Metz C., Phan B.: *PF_KEY Key Management API, Version 2.* July 1998.

[RFC2401] Kent S., Atkinson R.: *Security Architecture for the Internet Protocol.* November 1998.

[RFC2402] Kent S., Atkinson R.: *IP Authentication Header.* November 1998.

[RFC2403] Madson C., Glenn R.: *The Use of HMAC-MD5-96 within ESP and AH.* November 1998.

[RFC2404] Madson C., Glenn R.: *The Use of HMAC-SHA-1-96 within ESP and AH.* November 1998.

[RFC2406] Kent S., Atkinson R.: *IP Encapsulating Security Payload (ESP).* November 1998.

[RFC2408] Maughan S., Schertler M., Schneider M., Turner J.: *Internet Security Association and Key Management Protocol (ISAKMP).* November 1998.

[RFC2409] Harkins D., Carrel D.: *The Internet Key Exchange (IKE).* November 1998.

[RSQL03] Rouvroy G., Standaert F., Quisquater J., Legat J.: *Design Strategies and Modified Descriptions to Optimize Cipher FPGA Implementations: Fast and Compact Results for DES and Triple-Des.* Field-Programmable Logic and Applications (FPL), 2003, pp. 181–193.

[SLC02] Seng S., Luk W., Cheung P.: *Run-Time adaptive Flexible Instruction Processors.* Field-Programmable Logic and Applications (FPL), 2002, pp. 545–555.

[SMMS03] Saggese G., Mazzeo A., Mazzocca N., Strollo A.: *An FPGA-Based Performance Analysis of the Unrolling, Tiling, and Pipelining of the AES Algorithm.* Field-Programmable Logic and Applications (FPL), 2003, pp. 291–302.

[SRD03] Nazar A. Saqib, Rodrigues-Henriquez F., Diaz-Perez A.: *Two Approaches for a Single-Chip FPGA Implementation of an Encryptor/Decryptor AES Core.* Field-Programmable Logic and Applications (FPL), 2003, pp. 303–312.

[SSL2] Hickman, Kipp: *The SSL Protocol.* Netscape Communications Corp., Feb 9, 1995.

[SSL3] Frier A., Karlton P., Kocher P.: *The SSL 3.0 Protocol.* Netscape Communications Corp., Nov 18, 1996.

Passive network monitoring adapter intended for 10Gbps technology Passive network monitoring adapter intended for 10Gbps technology

Tomáš Martínek, Jan Kořenek, Jiří Novotný

E-mail: martinto@fit.vutbr.cz, korenek@fit.vutbr.cz, novotny@ics.muni.cz

Abstrakt

This paper presents an architecture of network monitoring adapter intended for 10Gbps technology. The proposed adapter allows producing statistics of input data flows, sampling packets with required properties, and payload checking of specific patterns. The architecture is designed for FPGA platform and is composed of several cooperating application specific nano-processors. As a preliminary implementation, 1Gbps version of monitoring adapter is presented.

1 Introduction

In recent years, the performance of computer networks has grown up very fast. The 1Gbps technology is replaced by the 10Gbps one. It becomes rather difficult for applications to analyze and process the data so fast. Therefore, the critical parts of applications are moved into the level of application specific circuits or programmable gate arrays, such as FPGAs.

Network traffic monitoring is typical application with high performance requirement. With 10Gps technology, up to 20 million packets arrive into the system per second. For each packet, it is necessary to

analyze control information in the header and to provide the packet classification. The classification process puts the packets into the specific categories based on requirements of the monitoring system. For some of categories, statistic information are produced and/or selected packets are sampled for more detailed analysis. The most critical part of monitoring adapter is the payload checking — the data of input packets must be analyzed for occurrence of specific patterns.

In this paper, we present architecture of monitoring adapter intended for 10Gbps technology. The adapter provides input packet classification and supports three techniques of packet sampling (deterministic, probabilistic and byte deterministic). Further, it produces two kind of statistic information — statistics based on packet length and statistics based on timestamps. Finally, it allows the payload checking of specified packets for containing up to 120 patterns 16 bytes long.

The paper is organized as follows. Section 2 describes architecture of 10Gbps monitoring adapter and its components. The performance limitations are described in Section 3 and implementation of preliminary 1Gbps version is presented in Section 4. Section 5 describes firmware design cycle. Conclusions are given in Section 6.

2 Adapter Architecture

The architecture of 10Gbps monitoring adapter is shown in Figure 1.

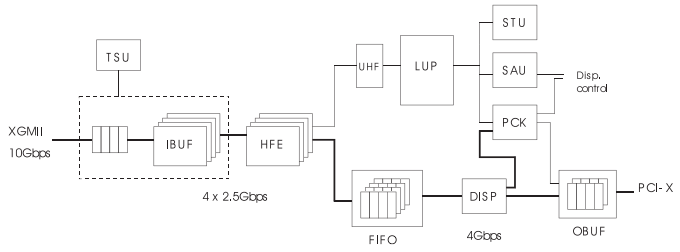


Figure 1 Architecture of 10Gbps adapter

The input packets arrive from the standard XGMII interface and they are evenly distributed into the four Input Buffers (IBUF), so that the 10Gbps input data stream is separated into four 2.5Gps streams. In

association with the packet data, the timestamp and identification information is stored into the IBUF. Timestamp is generated by Timestamp unit (TSU) and represents precise time information of packet arrival needed for statistic purposes. The IBUF is used as a packet storage and it also provides CRC computation checking.

Header Field Extractor (HFE) is a specific processor based on RISC architecture with instruction set dedicated for packet headers analyzing. The HFE reads packets from the Input Buffer, analyze control information included in headers and extracts those which are important for packet classification. This output data structure is called "Unified Header". The Unified Headers generated by four HFEs are stored in Unified Header Fifo (UHF) block and the data of original packets are continuously forwarded to the FIFO block.

Look-Up Processor (LUP) provides a classification of input packets. Unified Header structure is used as an input parameter of classification process and the output represents control word needed for further packet processing. Classification process is realized as a pass through a tree structure, where the leaf nodes represent control words and the pass is parameterized by UH. For implementation purposes, the fast associative memory TCAM is utilized for the first match. The rest of the classification process is controlled via program stored in addition SRAM memory. Content of TCAM and SRAM memories is dynamically generated by software tools.

Statistic Unit (STU) produces statistic information of up to 256 different types of packets. Information are separated into two parts — length and timestamp statistics. The length statistics contains: number of packets, sum of lengths of packets, sum of squares of packets lengths and min/max packet length. The timestamp statistics contains: sum of inter-packets intervals, sum of square of inter-packets intervals, min/max of inter-packet interval and timestamp of last packet. Architecture of STU is composed of processing unit and 256 sets of registers. The results are read and cleared by software driver.

Sampling Unit (SAU) produces information whether the packet is useful for some software application or not. The SAU contains up to 16 sampling cores and each one can be configured to do: probabilistic sampling — packet is passed through the unit with the probability 1/n; deterministic sampling — each n-th packet is passed through; and byte deterministic sampling — packet containing each n-th byte is passed

through. The packet can be sampled simultaneously by more then one core (given by control word from LUP) and if none of them needs the packet, it is discarded. The output information of SAU cores are propagated to the Dispatcher block.

Payload checker (PCK) is dedicated for fast pattern matching in a packet payload which is mostly important for intrusion detection. The TCAM memory is utilized to achieve 3.2 Gbps data throughput. Instead of matching one character per cycle, patterns are shifted past the CAM and a half word of CAM is always matched at once. Every pattern in CAM is marked by a number. The same type of number is included in the LUP result and specifies which patterns need to be matched for given packet. If any of patterns is matched, the packet is marked and sent to the SW for the next processing.

Dispatcher (DISP) block controls output data stream from FIFO block. Following situations are possible: If none of SAU and PCK needs a packet, it's discarded. If SAU or PCK needs a packet, it's forwarded to the Output Buffer (OBUF). If PCK needs a packet, it's also forwarded to the PCK and at the end of packet processing, the PCK has to confirm its validity in the OBUF. In addition to the packet data, DISP attaches the control information from STU, SAU and PCK.

Output Buffer (OBUF) is used as a storage for outgoing packets and it also controls output bus master data transfers from adapter to the software driver. It contains two queues — one for the data sampled by SAU block and the second one for the data intended for PCK block.

3 Performance Limitations

In the current architecture design, some performance critical places are known. At first, 10Gbps input data stream have to be evenly distributed into the Input Buffers. It causes, that some packets are processed out of order and it prevents SAU from correct sampling. This can be solved by simple Re-Order Unit, which reorder control words generated by LUP.

Next, LUP is designed to classify up to 8 million packets per second, but it's needed to classify up to 20 million packets per second to meet the worst case of 10Gbps technology. The LUP is able to classify more than 8 million packets per second, but there exists a trade-off between time for classification and complexity of classification process given by requirements of monitoring system.

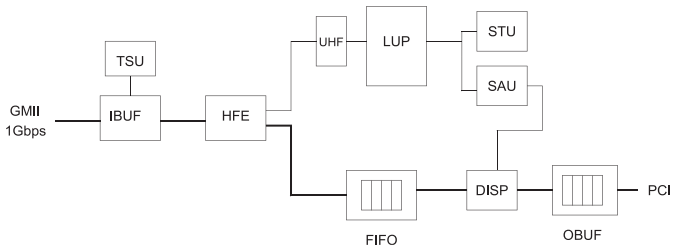


Figure 2 Architecture of 1Gbps adapter

Payload checking is generally the most performance critical part of the design. Bandwidth of payload checking strongly depends on number of patterns, length of patterns and available hardware resources. It's needed to find a trade-off between all these parameters. The current PCK architecture is designed to process 3Gbps data stream and it is able to search up to 120 patterns 16 bytes long. Only the first occurrence of arbitrary pattern is checked because of performance reasons — the remaining tests are left to the software application.

Although the PCI-X interface is assumed, only 4Gbps data bandwidth can be transferred to the software driver. According to this limitation, the output part of the adapter is designed to transfer variable length of packets. In this way, only packet headers can be transferred for packets which are not intended for payload checking.

4 Hardware Implementation

The implementation of 10Gbps monitoring adapter includes many issues in time critical operations and design complexity. For these reasons, the 1Gbps version of adapter has been developed as a preliminary version. In this way, we have also been able to check the major functionality and software interfaces of monitoring adapter.

The architecture of adapter preliminary version is shown in Figure 2. Relative to 10Gbps architecture, only one stream of packet processing is supported. Therefore only one instance of IBUF, HFE and FIFO blocks is included. The STU, SAU, LUP have exactly the same functionality, but only one LUP input interface is used. Dispatcher and Output Buffer are simplified because the PCK is not included and doesn't need any control.

Currently, the preliminary version of adapter has been implemented in the COMBO6 card [12] [10] [11] [13] developed under the Liberouter project [8] [9]. The basic functionality of adapter has been tested in hardware and is ready for performance tests. The implementation of final 10Gbps adapter started and should be finished soon because many components are exactly the same or similar in both designs.

5 Firmware Design Cycle

The Firmware Design Cycle is very similar to Software Development. The most used languages are VHDL and Verilog. The VHDL is preferred in academic world and in Europe's companies as the Verilog is used in USA and Japan. The VHDL was used in monitoring adapter development. We would like to explain design cycle on the example of 10Gbps monitoring adapter.

First, the definition of functionality was specified in whole team (application, software, firmware and hardware developers). Next, the functionality was split into several levels (software, firmware and hardware). On the firmware level, the architecture of building blocks and their interfaces were designed. As the algorithm implemented in firmware is very complex, the document by firmware developers was written and sent the rest of team for approval. The final document was assembled in several iterations together with the software developers and used for a primary implementation.

As the debugging and tests of the firmware is very complex in target hardware, the lot of simulation is needed on several level. After the algorithm is coded, the functional simulation proves the correctness of the design. This simulation doesn't include the physical restrictions (time limits) of target hardware. In next step, the synthesis, place and routing tools create VHDL model for timing simulation and transforms design into the bitstream, which can be downloaded into hardware (after the design was proved in timing simulations). Timing simulations considers the physical limits of the target hardware.

6 Conclusion

In this paper, we proposed the architecture of 10Gbps monitoring adapter. It is designed to measure statistics of incoming packets based on

packet length and arrival time. The filtration, sampling and fast matching in a packet payload is also supported.

Currently, the preliminary version of adapter has been implemented and basic functionality has been tested in hardware. Our future work concentrates to finish 10Gbps adapter and speed-up its time critical parts. In case of PCK block, it is possible to exploit reconfigurable computing based on FPGA [4] and speed-up pattern matching to achieve full 10Gbps speed. The usage of PCI Express technology is intended instead of PCI-X to increase output data transfer bandwidth.

7 Acknowledgements

This work was supported by the IST project SCAMPI [14] (IST-2001-32404) funded by the European Union. We are grateful to V. Smotlacha, who assembled the final version of firmware architecture document and P. Zemčík for his work devoted to architecture design and for his constructive comments.

References

[1] Antoš D., Kořenek J.: *String Matching for IPv6 Routers*. In: *SOFSEM 2004*, Matfyz Press, Praha 2004, pp. 205–210. ISBN 80-86732-19-3

[2] Antoš D., Kořenek J., Řehak V., Minařiková K.: *Packet header matching in Combo6 IPv6 router*. Technical Report 1/2003, CESNET, 2001.

[3] Antoš D., Řehák V., Kořenek J.: *Hardware Router's Lookup Machine and its Formal Verification*. In: *ICN'2004 Conference Proceedings*, volume II, University of Haute Alsace, Colmar, France, Gosier, Guadeloupe, French Caribbean 2004, pp. 1 002–1 007. ISBN 0-86341-325-0

[4] Baker Z. K., Prasanna V. K.: *Time and Area Efficient Pattern Matching on FPGAs*. In: *Twelfth ACM International Symposium on Field-Programmable Gate Arrays (FPGA 2004)*, February 2004, pp. 205–210.

[5] Baker Z. K., Prasanna V. K.: *Automatic Synthesis of Efficient Intrusion Detection Systems on FPGAs*. In: *Field-Programmable Logic and Applications, 14th International Conference (FPL 2004)*. to appear, September 2004.

[6] Clark C. R., Schimmel D. E.: *Efficient Reconfigurable Logic Circuits for Matching Complex Network Intrusion Detection Patterns*. In: *Field-Programmable Logic and Applications, 13th International Conference (FPL 2003)*, September 2003.

[7] Coppens J., Markatos E. P., Novotný J., Polychronakis M., Smotlacha V., Ubik S.: *SCAMPI — A Scaleable Monitoring Platform for the Internet*. In: *Proceedings of the 2nd International Workshop on Inter-Domain Performance and Simulation (IPS 2004)*, March 2004.

[8] Liberouter. *Liberouter Project WWW Page*. 2004 http://www.liberouter.org

[9] Novotný J., Fučík O., Antoš D.: *Project of IPv6 Router with FPGA Hardware Accelerator*. In: Peter Y. K. Cheung, George A. Constantinides, de Sousa J. T, editors, *Field-Programmable Logic and Applications, 13th International Conference (FPL 2003)*, vol. 2 778, Springer Verlag, September 2003, pp. 964–967.

[10] Novotný J., Fučík O., Bardas R.: *Schematics of COMBO-4MTX card*. Technical Report 13/2003, CESNET, 2002.

[11] Novotný J., Fučík O., Bardas R.: *Schematics of COMBO-4SFP card*. Technical Report 12/2003, CESNET, 2002.

[12] Novotný J., Fučík O., Kokotek R.: *Schematics and PCB of COMBO6 card*. Technical Report 14/2002, CESNET, 2002.

[13] Novotný J., Smotlacha V., Bardas R.: *Schematics of COMBO-PTM card*. Technical Report 15/2003, CESNET, 2003.

[14] SCAMPI. *Scampi Project WWW Page*. 2004. http://www.ist-scampi.org

[15] Sourdis I., Pnevmatikatos D.: *Pre-decoded CAMs for Efficient and High-Speed NIDS Pattern Matching*. In: *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2004)*, April 2004.

[16] Xilinx, Inc.: *Virtex-II 1.5V Field-Programable-Gate-Arrays*, November 2002. http://direct.xilinx.com/bvdocs/publications/ds031.pdf

[17] Xilinx, Inc.: *Virtex-II Pro Platform FPGAs*, March 2004. http://direct.xilinx.com/bvdocs/publications/ds083.pdf

Zálohování dat pomocí Open Source software

Oldřich Balák

E-mail: obal@civ.zcu.cz

Spolu s výměnou zálohovacího hardware na naší univerzitě vznikl požadavek na obměnu zálohovacího software a sjednocení zálohování různých systémů.

Hlavní požadavky, které jsme na zálohovací software měli :

- hlavní zálohovací server běží na Linuxu – v našem případě Debian,
- klienti pro různé platformy – Unix, Linux, Windows,
- možnost zálohovat AFS,
- schopnost zálohovat na pásku (páskového robota, včetně jeho obsluhy) a do souborů.

Vzhledem k cenám komerčních zálohovacích softwarů jsme se rozhodli zkusit cestu Open Source software. Zpočátku jsme zkoušeli software s názvem Amanda, který se zdál být vhodný pro naše potřeby, nicméně pak jsme objevili software Bacula a ten svými vlastnostmi výrazně Amandu převyšuje.

Stručný popis sytému Bacula

Celý zálohovací systém Bacula se skládá ze tří základních démonů, několika dalších pomocných programů a může být doplněn pomocnými skripty.

bacula directorje hlavní démon, instaluje se na zálohovací server. Stará se o chod celého systému a v jeho konfiguraci se nastavuje způsob zálohování jednotlivých klientů, tj. co, kdy a kam zálohovat.

bacula-fd je démon, který běží na klientovi a na požadavek bacula-directoru odesílá data pro zálohování.

bacula-storage je démon, který se stará o zápis dat zálohovací media (pásky nebo disky).

bacula console je nástroj pro administraci systému. Umožňuje zjišťovat stav systému, spouštět, případně zastavovat jednotlivé úlohy, recyklovat zálohovací media a pod. Může být nainstalován na samostatném počítači a existuje pro většinu platforem, jako klient. V případě Linuxu je ve třech provedeních jednom textovém – bconsole a dvou grafických – gnomeconsole a wxconsole, které však zatím obsahují drobné chyby.

databáze SQL – veškeré informace o systému jsou ukládány v SQL databázi, to znamená konfiguraci, provedené zálohy a pod. Bacula umožňuje použití SQLite, MySQL nebo PostgeSQL.

utilita programy pro obsluhu pásek a robotů. Jsou spouštěny ze skriptů, takže lze použít téměř cokoliv. Bacula má ale připravenou konfiguraci pro **mtx** a **mt**.

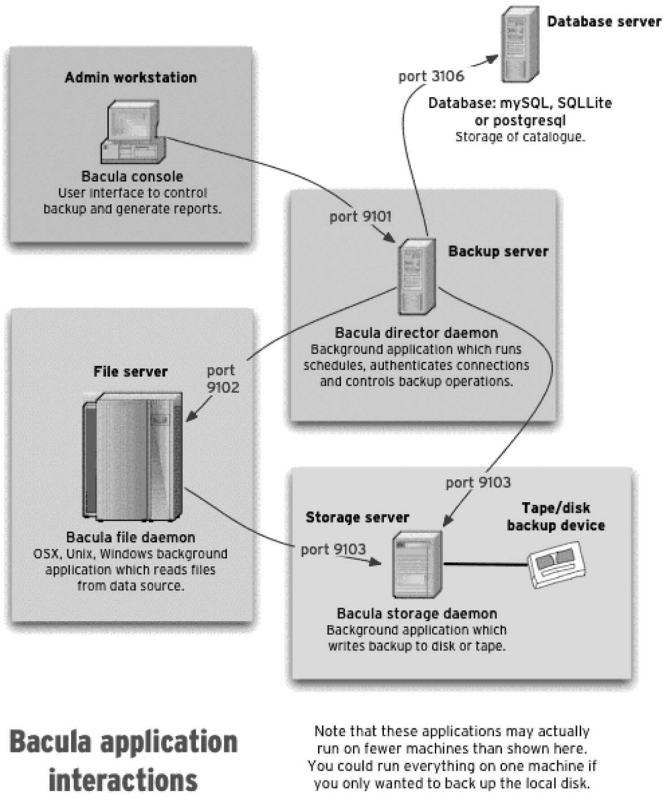
Jednotlivé části systému mohou být nainstalovány na různých strojích. Pro vzájemnou komunikaci tři, respektive čtyři porty. Standardně se používají porty 9101 pro připojení bacula console pro administraci systému, port 9102 pro komunikaci s klientem a port 9103 pro komunikaci se serverem obsluhujícím zálohovací zařízení (páska, disk). Pokud se používá databáze na samostatném stroji, využívá se ještě další port.

Vše zobrazuje obrázek 1.

U nás použitý hardware je:

Hlavní zálohovací server s procesorem Intel Pentium 4 na 3.2 GHz, ke kterému je připojené diskové pole z S-ATA disků, pásková knihovna Exabyte Magnum 20 s dvěma mechanikami IBM LTO2. Knihovna má 143 pozic pro pásky, ty mají kapacitu 200 GB v nativu. Pro zálohování na pásky používáme kompresi mechanik . Pásky uvádějí 400 GB s kompresí. Zkušenost je např. při zálohování AFS volumů 300 až 330 GB na pásku.

Operační systém serveru je Linux Debian a jádrem 2.4.26. Je na něm nainstalován bacula-director, bacula-storage, bacula console a také bacula-fd, aby mohl zálohovat sám sebe. Bacula na serveru je verze 1.34.5.



Obr. 1

Na klientech máme odzkoušené systémy Linux, Windows NT, 2000 a XP, Digital UNIX V4.0D, IRIX, Sun OS v. 5.6 a 5.9. Bacula je ve verzích od 1.32f po 1.35 podle operačního systému.

Některé vlastnosti

- zálohování na pásky a do souborů.
- spooling na disk před zálohováním na pásku, to je důležité pro životnost pásek, páskám nejvíc vyhovuje souvislý zápis, přerušovaný zápis životnost výrazně snižuje.

- možnost spouštění skriptů před a po zálohování, jak na straně klienta, tak na straně serveru, využití při zálohování AFS, LotusNotes, Windows, SQL databází a pod.
- umožňuje pokračovat na pásce a rozdělit velkou zálohu na několik pásek (souborů) – u Open Source software ne vždy obvyklé
- priorita procesů – to umožňuje např, aby záloha databáze proběhla až skončí všechny běžné zálohovací procesy.
- schopnost automatické recyklace pásek a souborů.
- kontrola konfiguračních souborů.
- velmi dobrá dokumentace.
- je to živý system, neustále se vyvíjí.
- nastavení zálohování se provádí pro všechny klienty na serveru.
- některé nastavení se dají (musí) nastavit na několika místech – někdy nepřehledné, zbytečně složité.
- nutnost restartovat director démona při většině změn (některé se dají udělat za chodu).

Specifika pro určité systémy nebo jejich části

- Linuxové a Unixové stroje na úrovni systému – většinou není problém.
- Windows (NT, W2K, XP) . U Windows nejdou některé soubory za běhu systému běžným způsobem přečíst (např. soubory registru). Je proto nutné vytvořit sktipt, který provede NTBackup pro systémové nastavení (na to je v NTBackupu volba) a případně pro další soubory, které Bacula není schopna přímo zálohovat.
- AFS. Pro zálohování AFS se požívá skript, který pro vybranou skupinu volumů provede vos dump na diskové pole (ten spouští přímo Bacula). Je to z důvodu zachování atributů pro jednotlivé soubory (liší se od unixového souborového systému). Vos dump uloží kompletní volumy do samostatných souborů včetně atributů. Tyto soubory jsou pak Baculou nahrány na pásky.

- Databáze a jim podobné záležitosti (např. Lotus Domino). Opět je nutné před vlastní zálohou pomocí baculy spustit skript, který provede zálohu na úrovni aplikace a tu pak už zálohovat běžným způsobem.

Stručný přehled konfigurace systému

Konfigurace se provádí pomocí přehledných konfiguračních souborů. Nejjednodušší je konfigurace klienta, kde se konfiguruje vlastně jen přístup k řídícímu serveru.

```
# Bacula File Daemon Configuration file

Director {
    Name = server-dir
    Password = "heslo-klient"
}

FileDaemon {
    Name = klient-fd
    FDport = 9102
    WorkingDirectory = /var/lib/bacula
    Pid Directory = /var/run/bacula
}

Messages {
    Name = Standard
    director = paskos-dir = all, !skipped
}
```

Dále je nutné zkonfigurovat páskového démona:

```
# Bacula Storage Daemon Configuration file

Storage {
    Name = server-sd
    SDPort = 9103
    WorkingDirectory = "/var/lib/bacula"
    Pid Directory = "/var/run/bacula"
    Maximum Concurrent Jobs = 5
}
```

```
Director {
    Name = server-dir
    Password = "heslo-paska"
}

Device {
    Name = File
    Media Type = File
    Archive Device = /mnt/backup
    LabelMedia = yes;
    AutomaticMount = yes;
    RemovableMedia = no;
    AlwaysOpen = no;
}

Device {
    Name = Magnum20
    Media Type = LTO2
    Archive Device = /dev/tape0
    Changer Device = /dev/robot
    Changer Command = "/etc/bacula/scripts/mtx-changer
        %c %o %S %a %d"
    Autochanger = yes
    Drive Index = 0;
    AutomaticMount = yes;
    Removable Media = yes;
    Spool Directory = /mnt/backup/spool/tape0
    Maximum Spool Size = 100gb
}
```

Ostatní konfigurace se odehrává na serveru a directorem.

```
# Bacula Director Configuration file

Director {
    Name = server-dir
    DIRport = 9101
    QueryFile = "/etc/bacula/scripts/query.sql"
    WorkingDirectory = "/var/lib/bacula"
    PidDirectory = "/var/run/bacula"
```

```
    Maximum Concurrent Jobs = 5
    Password = "heslo-server"
    Messages = Standard
}

Catalog {
    Name = MyCatalog
    dbname = bacula; DB Address = localhost; DB Address = localhost;
        user = bacula; password = "heslo-sql"
}

Messages {
    Name = Standard
    mailcommand = "/usr/lib/bacula/bsmtp -h mail.zcu.cz -f \"\(Bacula\)
        %r\" -s \"Bacula: %t %e of %c %l\" %r"
    operatorcommand = "/usr/lib/bacula/bsmtp -h mail.zcu.cz -f \"\(
        Bacula\) %r\" -s \"Bacula: Intervention needed for %j\" %r"
    mail = komu@zcu.cz = all, !skipped
    operator = komu@zcu.cz = mount
    console = all, !skipped, !saved
    append = "/var/lib/bacula/log" = all, !skipped
}

Storage {
    Name = File
    Address = 147.228.XXX.XXX
    SDPort = 9103
    Password = "heslo-paska"
    Device = File
    Media Type = File
}

Storage {
    Name = Magnum20_1
    Address = 147.228.XXX.XXX
    SDPort = 9103
    Password = "heslo-paska"
    Device = Magnum20
    Autochanger = yes
```

```
    Media Type = LTO2
}

Client {
    Name = klient-fd
    Address = 147.228.XXX.XXX
    FDPort = 9102
    Catalog = MyCatalog
    Password = "heslo-klient"
    File Retention = 30 days
    Job Retention = 30 days
    AutoPrune = yes
}

FileSet {
    Name = "Fileset1"
    Include {
        Options {
            signature=MD5
            compression=GZIP
        }
        File = /
        File = /var
    }
}

FileSet {
    Name = "Fileset2"
    Include {
        Options {
            signature=MD5
        }
        File = "c:/"
        File = "d:/"
    }
    Exclude {
        File = "c:/Documents and Settings/*"
        File = "c:/pagefile.sys"
```

```
    }
}

Schedule {
    Name = "Mesicni"
    Run = Level=Full 1st sun at 1:00
    Run = Level=Differential 2nd-5th sun at 1:00
    Run = Level=Incremental mon-sat at 1:00
}

Pool {
    Name = Zaloha_tape
    Pool Type = Backup
    Recycle = yes
    AutoPrune = yes
    Volume Retention = 30 days
    Cleaning Prefix = CLN
}

Job {
    Name = "Zaloha"
    Type = Backup
    Client=klient-fd
    FileSet="Fileset1"
    Schedule = "Mesicni"
    Storage = Magnum20
    Messages = Standard
    Pool = Zaloha_tape
    RunBeforeJob = "/etc/bacula/scripts/pomocny_skript_pred"
    RunAfterJob = "/etc/bacula/scripts/pomocny_skript_po"
    Write Bootstrap = "/var/lib/bacula/Klient.bsr"
    Priority = 10
}
```

Z jednotlivých částí je zřejmé, čeho se týkají. Nastavení v sekcích klientů a pásek musí korespondovat s odpovídajícím nastavením démonů klientů a pásek. Uvedený příklad je samozřejmě velmi jednoduchý a slouží pouze jako ukázka, ale správnou kombinací jednotlivých parametrů se dá docílit komplikovaného, přesto funkčního nastavení.

Porovnání některých zálohovacích systémů

Pokud srovnáme software Bacula a Legato Networker, zjistíme, že jsou si svojí koncepcí, možností konfigurace, obsluhovanými zařízeními a operačními systémy podobné. Hlavní rozdíly jsou v možnosti ovládat nastavení, zálohy a obnovu na klientovi pouze pro něj. Bacula se konfiguruje společně pro všechny klienty na serveru a neumožňuje ovládat úlohy pro jednoho klienta samostatně. Je sice možné mít nainstalovanou konzoli na libovolném počítači, vždy jsou však zobrazeny všechny informace o všech klientech a úlohách a je možné provádět operace (záloha, obnova, ale i mazání a pod.) pro všechny klienty. Nelze tedy umožnit přístup uživateli na straně klienta pouze k jeho záloze, její konfiguraci a umožnit mu obnovu pouze jeho dat. Dále je to absence ekvivalentu Group Control u Baculy proti Networkeru. Každá úloha má svojí vlastní definici co, kdy a kam budeme zálohovat. Samozřejmě může být pro jednoho klienta definováno více úloh s různými parametry. Pokud, ale chceme spustit mimo pořadí několik úloh, musíme je spustit samostatně. Další drobnou nevýhodou je již zmíněná nutnost restartovat director démona při změnách konfigurace. Poslední nevýhodou je absence kvalitního grafického prostředí (i když to je diskutabilní a někdo možná grafikou pohrdne). Existují gnomeconsole, ta je v novějších verzích nahrazena wxconsolí. S oběma se pracuje obdobně jako s textovou konzolí. Wxconsole přináší možnost nastavení obnovy ze záloh interaktivním způsobem, zatím však obsahuje řadu chyb a způsob ovládání je v některých okamžicích „nešikovný“ a u většiny voleb není cesta zpět, takže nezbývá, než konzolu ukončit a začít znovu. Krátce se zmíním o software Amanda, který jsem také zkoušel (poznatky se vztahují k březnu až květnu 2004, kdy jsem Amandu zkoušel). Na rozdíl od Baculy (o Networkeru nemluvě), která má velmi rozsáhlou a dobře napsanou dokumentaci, k Amandě neexistuje téměř žádná a pokud ano, obsahuje chyby nebo nepřesnosti dané předbíháním vývoje před dopisováním dokumentace. Hlavní nedostatky byly, ale použití nové pásky pro každou úlohu a naopak neschopnost rozdělit velkou úlohu na víc pásek.

Shrnutí

Přes některé drobné nedostatky dané jednak chybami programu (ty jsou plynule odstraňovány) a jednak koncepcí (s tím je nutno se smířit) se jedná o velmi zajímavý a životaschopný zálohovací systém, za který není

nutné platit „velké“ peníze. Pokud jste zvyklí na systém Legato Networker nečiní přechod na Baculu žádné zvláštní problémy.

Literatura

[Bacula] http://www.bacula.org

[OpenAFS] http://www.openafs.org

[Legato Networker] http://www.legato.com

[Amanda] http://www.amanda.org

[MTX] http://mtx.badtux.net

Ochrana proti škodlivému SW

Pavel Baudiš

E-mail: baudis@asw.cz

Motto:
A to se nebojíte nechat ty viry v počítači přes noc bez dozoru?
(dotaz mladé nastupující sekretářky)

Škodlivý kód

Náš svět bohužel není ideální – vidíme to všude kolem nás a samozřejmě se to týká i světa počítačů. Proto se už od jejich prvního masovějšího rozvoje objevují programy, jejichž cílem není uživatelům pomáhat, ale naopak škodit, či jejich počítače nějakým způsobem zneužívat. Takové škodlivé programy se anglicky nazývají *malware* a dají se rozděliit do mnoha kategorií. Snad nejznámější z nich jsou *počítačové viry*.

První počítačový virus vytvořil na začátku osmdesátých let Fred Cohen. Pro svoje pokusy využíval izolované počítače s operačním systémem Unix a kromě praktických pokusů vytvořil i formální matematickou definici počítačových virů. Prokázal také, že není možno vytvořit program, který by o jakémkoli jiném programu v konečném čase dokázal stoprocentně říci, zda je či není počítačovým virem.

I když definic počítačového viru existuje celá řada, je zde situace ještě mnohem jednodušší než u jiných škodlivých programů. Za virus je možno označit program, který je schopen se šířit a množit, ať už v rámci jednoho počítače nebo v počítačové síti. Speciální kategorii tvoří počítačové červy, které se šíří právě v sítích a nevyžadují žádné hostitelské programy. Virus je program, a jako takový je nutno jej spustit, aby mohl být vůbec aktivní a aby se mohl dále množit. Proto viry využívají vektory šíření – a tyto vektory pak rozhodují o tom, zda se jedná o teoretickou možnost nebo reálnou hrozbu. Krásně je to vidět i v minulosti: první reálné viry se objevily s velkým využitím počítačů kompatibilních

s IBM PC a s operačním systémem MS-DOS. Tehdy se pro výměnu dat a programů využívaly v naprosté většině případů pouze diskety, a proto zažily takový úspěch boot viry. K tomu samozřejmě přispěla i problematická koncepce pořadí diskových jednotek při zavádění systému, kdy měla (a dodnes často má) přednost právě disketová jednotka. Na disketách se často přenášely i programy – proto byly velmi úspěšné souborové viry, které byly velmi krátké (většinou 1 až 5 kB) a jednoduché.

Velký zlom nastal až s příchodem operačního systému Windows 95, který kromě multitaskingu a grafického rozhraní přinesl i masové nasazení kancelářských programů MS-Office. A právě zcela chybná koncepce maker v těchto programech (zejména MS-Word a MS-Excel) přinesla obrovskou vlnu nového druhu virů – makrovirů. Makra jako taková mohou být bezesporu užitečná, nicméně firmě Microsoft se podařilo implementovat je tak, že vytvořila ideální prostředí pro počítačové viry: provádění maker není možno vypnout, existují automaticky prováděná makra, spouštěná například při spuštění aplikace či při otevření dokumentu, ale makro je možno přiřadit i položce v menu či stisknuté klávese, přičemž makra z dokumentu mají přednost před makry v systému. Makroviry dokázaly tyto možnosti využít opravdu na doraz, jejich výhodou bylo i to, že uživatelé sdílejí daleko víc dokumenty než programy. Makroviry také nabouraly teorii o tom, že nebezpečné jsou pouze programy, zatímco výměna dokumentů je zcela bezpečná. Je pravda, že sdílení pouhých dat nemůže spustit jakýkoli škodlivý program, problém je ale v tom, že dokumenty dnes neobsahují pouze čistá data, ale také programy – právě makra.

V březnu 1999 se objevil makrovirus Melissa.A, který pro svoje šíření začal aktivně využívat elektronickou poštu – už jen pasívně nečekal, až jej uživatel někomu odešle, ale sám sebe poslal na padesát adres, které našel v adresáři programu MS-Outlook. Tato metoda byla natolik úspěšná, že ji brzo začali používat i další autoři virů – a od té doby se elektronická pošta stala nejrozšířenějším kanálem pro šíření virů.

Makroviry jsou velice zajímavé i z hlediska jejich vytváření – jsou psány ve vyšším programovacím jazyce (např. VBA – Visual Basic for Applications) a šíří se včetně zdrojového kódu, takže je velmi jednoduché je změnit. Přesto mají jednu zásadní nevýhodu – mohou být aktivní pouze tehdy, když běží jejich hostitelská aplikace, a jejich možnosti jsou příliš spojeny s tím, co jim tato aplikace umožňuje. Již v roce 1996 se objevil první virus, vytvořený přímo pro operační systém Windows, který

tyto problémy odstraňoval. Ten ale neměl reálnou šanci se jakkoli rozšířit a autorům takových virů to trvalo několik let, než se naučili plně využívat všech možností, které jim operační systémy Windows nabízely, a také toho, že stále více počítačů je připojeno k Internetu. Dnes jsou však tyto programy největším nebezpečím a také nejrozšířenějšími škodlivými programy vůbec.

Ještě než se jimi budeme zabývat podrobněji, chtěl bych se zmínit i o další kategorii virů, které po několik měsíců zaznamenávaly velké úspěchy právě díky příznivému vektoru šíření. Jedná se o skript viry, z nichž nejznámější jsou určitě dva: Loveletter, který 4. května 2000 vyvolal nečekanou a masivní epidemii a který byl určitě nejúspěšnějším skriptovým virem, a dále VBSWG.J, mnohem známější pod jménem Kournikova, který se také v únoru 2001 nakrátko velmi rozšířil, protože sliboval obrázek známé tenistky. Zajímavé je, že oba autoři těchto virů byli dopadeni a že oba pravděpodobně řádově podcenili schopnosti šíření svého výtvoru.

Vraťme se ale zpět k nativním virům Windows. Řada z nich ještě fungovala klasickým způsobem – nevyužívala elektronickou poštu a připojení k Internetu. Nejznámější je určitě CIH (Černobyl), který způsobil 26. dubna 1999 velké škody hlavně v Asii, když přepsal FlashBIOS na základní desce mnoha počítačů. Šířil se hlavně s některými drivery a pirátskými kopiemi Windows 95. Ale už koncem roku 1998 zaznamenal velký úspěch virus Ska (Happy99), který se připojoval ke každé odcházející zprávě a kromě zobrazení ohňostroje na obrazovce měnil jeden systémový program. Od té doby se viry tohoto typu objevují zcela pravidelně a jejich šíření je stále prudší a mohutnější. Dnes je bohužel vypuknutí celosvětové epidemie otázkou několika málo hodin. Jako příklad nejúspěšnějších virů roku 2000 až 2002 jmenujme Sircam, Nimda či Klez.

Kromě počítačových virů existují i další škodlivé programy. Jejich definice je ale mnohem těžší – a vlastně nemožná. Vždyť to, co někdo považuje za škodlivý program, může být pro jiného užitečným nástrojem nebo dobrým žertíkem, a tak zde často existují šedé oblasti, kdy se mluví spíše o „potencionálně nebezpečných aplikacích“. Většina škodlivých programů, které nejsou schopny se samy šířit, bývá označována jako *trojské koně*. Je to proto, že (podobně jako ve starověké Troji) ve skutečnosti provádějí něco jiného, než nabízejí. Typickým příkladem může být „výborná hra“, která ale smaže obsah disku. Trojské koně existovaly

už dávno, ale právě kvůli chybějícímu mechanismu šíření nepředstavovaly takové nebezpečí jako nyní. Dnes je možno je stáhnout z Internetu, z diskusních skupin Usenet či IRC a v nezanedbatelné míře i přes viry – řada dnešních virů totiž do infikovaných počítačů nějakého trojského koně instaluje. Trojské koně můžeme rozdělit do několika skupin: backdoory (zadní vrátka) umožňují vzdálený přístup a ovládání počítače, keyloggery slouží ke zjišťování přístupových hesel či PINů kreditních karet a dialery k volání na žluté (tj. draze placené) telefonní linky. Všechny tyto programy jsou velice nepříjemné a mohou vést k velkým problémům a finančním ztrátám. Zadními vrátky kompromitované počítače mohou sloužit například k získávání utajovaných informací či k útokům typu DDoS. S dialery je občas zase legální problém s detekcí, kdy jejich autoři tvrdí, že se jedná o legální obchod prováděný se souhlasem uživatele.

Dnešní situace je celkem jasná – hlavním zdrojem pro šíření virů jsou dva kanály (vektory šíření): elektronická pošta a bezpečnostní díry v operačních systémech či aplikacích. Je evidentní, že k úspěchu virů přispívá i zcela jasná monokultura v oblasti OS, aplikací, internetového prohlížeče či klienta elektronické pošty. Velmi významné je i sofistikované sociální inženýrství, které autoři virů využívají k přinucení uživatelů k aktivní spolupráci při šíření virů (viz výše uvedené skript viry, ale i další nabízení zajímavých informací, obrázků či dokumentů nebo naopak informací o zrušení účtu, inkasování peněz z karty a podobně).

Demonstrujme si výše uvedené tvrzení na konkrétních příkladech z roku 2003 a 2004. V těchto letech došlo k dosud nejmasovějším epidemiím virů šířících se elektronickou poštou. Známé jsou například varianty viru Sobig (leden až září 2003), z nichž zejména Sobig.F, který se posílal v mnoha kopiích, způsobil kolaps mnoha poštovních serverů. Všechny varianty (až na tu první) měly časově omezené šíření a do systému instalovaly trojského koně, který je využíván pro masivní šíření spamu a prvotní nástup nových variant. Nebezpečné může být i odesílání existujících zpráv cizím uživatelům tak, jako to dělá virus Bugbear.B (a předtím Bugbear či Sircam). Může totiž dojít k velmi nepříjemnému úniku citlivých informací. Největší epidemii vyvolal v lednu letošního roku virus Mydoom.A, který překonal všechny rekordy a kromě jiného z Internetu na 13 dní vymazal server http://www.sco.com. V první polovině letošního roku pak došlo i k zajímavé válce autorů virů – skupin, které stojí ze viry Beagle, Netsky a Mydoom. Ty jednak likvidovaly „konkurenční“ viry a jednak obsahovaly posměšné vzkazy protivníkům.

Během krátké doby se objevily desítky variant, které reagovaly na akce druhé strany. V menší míře tato válka trvá dodnes, i když jeden z hlavních autorů virů Netsky byl v květnu zadržen v Německu.

Řada virů (a červů) se ale vůbec nešíří elektronickou poštou – podobně jako kdysi velmi známý Morrisův worm (listopad 1988) využívají slabiny a bezpečnostní díry OS a aplikací a pro své šíření ani nepotřebují spolupracujícího uživatele. Jimi vyvolané epidemie jsou proto ještě mnohem rychlejší – může se jednat o desítky minut! Prvním „novodobým“ červem byl Codered, využívající díru v MS IIS WEB Serveru. Ten během krátké doby napadl kolem 300 000 počítačů a pokoušel se o DDoS útok na server Bílého domu. Je zajímavý tím, že na napadeném počítači existuje pouze v operační paměti a nemění nic na pevném disku. Velmi podobný byl SQLSlammer z ledna 2003 – tento velmi krátký červ dokázal vyřadit z chodu celé segmenty Internetu a způsobit globální problémy. Je zajímavé, že opravná záplata existovala poměrně dlouho, ale řada administrátorů ji ignorovala. Vloni v srpnu se objevil červ Blaster, který využíval chybu DCOM/RPC. Ta byla objevena o necelý měsíc dříve a i na ni již existovala záplata, nicméně se odhaduje, že červ nakazil více než milión počítačů po celém světě. Naštěstí se jeho přítomnost ve většině případů projevovala vynuceným restartem počítače, a tak bylo jeho odhalení a případné odstranění usnadněno. Zatím poslední velká epidemie, způsobená červem, pochází z přelomu dobna a května letošního roku. Šlo o virus Sasser, zneužívající bezpečnostní díru známou jako LSASS. Oprava této díry byla k dispozici od poloviny dubna, ale podobně jako u viru Blaster se jejím odstraněním většina uživatelů nezabývala. Červ se projevuje zpomalením počítače a restarty, problémy způsobil nejen domácím uživatelům ale i velkým firmám – například několika leteckým společnostem a bankám, poště na Tchajwanu či dokonce Evropské komisi. Čerstvě osmnáctiletý autor tohoto červa (a také virů Netsky) byl sedmého května zadržen v Německu.

Budoucnost nevypadá příliš růžově. Podle mnoha odhadů se dnes na světě vyskytují skoro dva milióny počítačů, které obsahují nějakého trojského koně typu backdoor a které tedy mohou být zneužity nejen k šíření spamu, ale i k DDoS útokům na libovolný cíl a také k rozesílání nových virů. Nové bezpečnostní díry v nejpoužívanějších programech se objevují takřka denně a přitom uživatelé jejich záplatování vesměs ignorují. Je otázkou, kolik toho změní nová aktualizační politika firmy Microsoft, uvedená s balíčkem SP2 (implicitně zcela automatické aktualizace) a in-

tegrovaný firewall. Obávám se, že i kdyby všichni uživatelé Windows XP tyto technologie používali (což je nereálné), zůstává velká armáda počítačů se staršími systémy, které zůstanou nechráněny. A jak jsme viděli již mnohokrát, nejslabším článkem je vždy uživatel, a metody sociálního inženýrství tak mohou slavit velké úspěchy.

... A OCHRANA PROTI NĚMU

Jak se tedy proti všem výše zmíněným hrozbám a nebezpečím bránit? Ideální by bylo spolehlivé stoprocentně funkční řešení, které by bylo natolik obecné, že by nepotřebovalo žádné aktualizace a fungovalo by jednou pro vždy. Jak už jsme si ale řekli, už Fred Cohen dokázal, že teoretické řešení neexistuje, obávám se však, že neexistuje ani dostatečně obecné řešení, které by bylo v praxi použitelné.

Základní dnes používanou ochranou je *detekce známých druhů* virů a škodlivých programů. Ta je celkem jednoduchá a logická a má jak řadu výhod, tak nevýhod. Výhodou je to, že uživatelé jí rozumějí a vědí, co od ní mohou čekat. Je spolehlivá a dokáže škodlivý kód rozeznat dříve, než je spuštěn. Nevýhodou je samozřejmě to, že škodlivý program musí být znám dříve, než je detekce přidána. To vyžaduje velmi častou aktualizaci (dříve třeba měsíčně, dnes i několikrát denně) a znamená, že pokud není detekce včas přidána, škodlivý kód není detekován a může být propuštěn či aktivován. Z těchto důvodů hledají antivirové firmy obecnější řešení, jako je například generická detekce (detekce celých rodin či skupin virů) nebo heuristika (analýza kódu a následné rozhodování, zda je škodlivý či nikoliv). Tyto metody mají částečné úspěchy, ale nejsou všelékem a existují způsoby, jak je obelstít. Zejména heuristika je proti dnešním, ve vyšších programovacích jazycích psaným virům většinou neúčinná. Hledání známých druhů virů dnes většinou funguje buď při přístupu (on access), kdy je testován veškerý kód, který má být v systému spuštěn, nebo na vyžádání (on demand), kdy je spuštěno dávkové prohlížení souborů. Je jasné, že testování při přístupu je mnohem účinější a spolehlivější – funguje totiž neustále. Představitelem detekce známých druhů virů je klasický antivirový program.

Druhou metodou, která je používána v boji proti škodlivému kódu, *je monitorování činnosti systému.* Výhodou tohoto přístupu je to, že je poměrně obecný a nevyžaduje neustálé aktualizace. Nevýhodou je, že často funguje až v době, kdy už je škodlivý program aktivní, a tak

nedokáže předejít jeho spuštění a případné další činnosti (včetně vyřazení této ochrany z činnosti). Takto funguje například firewall.

Poslední obecnou metodou je *hlídání integrity dat*, kdy program monitoruje, zda nedošlo v programech či v systému k nějakým důležitým změnám. Výhodou je to, že je schopen změny spolehlivě zjistit a často i uvést programy do původního stavu, nevýhodou pak to, že tato metoda funguje ex post – tedy až poté, co škodlivý program provedl svoji činnost.

Je jasné, že nejúčinnější ochranou je zkombinování všech dostupných prostředků a metod pro ochranu. Každá překážka, postavená škodlivému programu do cesty, snižuje jeho šanci na úspěch. Kromě technických řešení je ale stejně důležitá prevence – tedy soubor organizačních opatření, které by v nějaké míře měly platit od největších firem až po domácí uživatele.

Ve firmách je potřeba uživatelům vysvětlit, jakým způsobem se chovat, co smějí, co nesmějí a proč. Je vhodné na poštovních serverech zakázat nebezpečné přílohy, a tak opravdu radikálně snížit nebezpečí infekce elektronickou poštou. Je potřeba včas aplikovat všechny bezpečnostní záplaty, důležitá data pravidelně zálohovat a mít vypracován krizový scénář, aby v okamžiku vypuknutí problémů každý věděl, co má dělat. Je potřeba nasadit vhodný antivirový program, který nabízí centrální správu a automatické aktualizace bez velkého zatížení provozu. Je potřeba nastavit firemní firewall tak, aby umožňoval pouze to, co je pro chod firmy potřeba.

I domácí uživatelé by měli svůj počítač chránit. Dnes existuje několik kvalitních antivirových programů, které jsou pro domácí použití zdarma, stejně tak by mělo být samozřejmostí nasazení a používání osobního firewallu a aplikace záplat. Vždyť podle posledních průzkumů vydrží počítač s implicitní instalací Windows XP v čistém stavu méně než dvacet minut!

Práce na antivirovém programu je časově i technicky velice náročná. Nejde jen o implementaci antivirového engine do nejrůznějších operačních systémů, poštovních serverů, firewallů a podobně. Velmi složité je i udržování virových definic v aktuálním stavu. Vždyť měsíčně se dnes objeví více než 2000 škodlivých programů. Ne všechny se sice vyskytují přímo u uživatelů, nicméně detekovat je potřeba všechny. Zkrátka v případě antivirových programů se jedná spíše o službu, která vyžaduje denní náročnou práci, než o produkt. Právě proto se v oblasti otevře-

ných systémů příliš mnoho kvalitních antivirových programů neobjevilo. Možná je to i tím, že tradiční antivirové firmy spolu po technické stránce zejména v oblasti nových virů (rychlá detekce, sledování šíření, pojmenovávání) poměrně úzce spolupracují. Bez této spolupráce by reakce na nové hrozby nemohla být tak rychlá.

Firewally v praxi

Josef Pojsl

E-mail: jp@tns.cz

Abstrakt

Pro zařízení oddělující sítě s různou úrovní důvěryhodnosti se vžil termín firewall. Pod tímto názvem se skrývá několik technologií, které se mohou různě kombinovat. Jejich základní povaha a principy jsou všeobecně známy.

V praxi se setkáváme s velmi různou znalostí problematiky. Zásluhou propagace v 90. letech 20. století se rozšířil názor, že síť každé organizace má být chráněna firewallem. Představy o tom, co to vlastně znamená, se však diametrálně rozcházejí.

Příspěvek se zaměřuje na praktické zkušenosti s instalací a konfigurací firewallů ve státním i soukromém sektoru v ČR. Shrnuje poptávku a očekávání zákazníků (IT managementu) od firewallu a mapuje rozdíl mezi těmito očekáváními a funkcemi, které může moderní firewall skutečně nabídnout.

Při instalaci a svěřené správě firewallů a dalších síťových prvků se setkáváme s požadavky nejrůznějšího druhu. Zadávají je zejména správci podnikových sítí, někdy přímo vedoucí pracovníci IT oddělení našich zákazníků, kterými jsou střední a větší organizace v soukromé i rozpočtové sféře.

Zprvu jsem považoval různé mýty a pověry, nesprávné předpoklady i nesplnitelná očekávání spojená s firewally za kuriozity. Některé z nich se však opakují s tak železnou pravidelností, že se staly jistým druhem folklóru. Navíc jsem přesvědčen, že o situaci na poli bezpečnosti informačních systémů u nás ledacos vypovídají, proto jsem se rozhodl o ně podělit s účastníky XXV. konference EurOpen.CZ.

1 Gigabitový firewall aneb Rychlost nade vše

V podniku se staví gigabitová síť, i firewall proto musí zvládnout 1 Gbps! Zní to logicky, není-liž pravda?

Připojení do Internetu má však kapacitu 10 Mbps, a přestože firewall zprostředkovává i komunikaci mezi veřejnými servery (zapojenými na zvláštní síťové kartě firewallu) a interní sítí, špičkový tok na této „gigabitové“ síti zdaleka nedosahuje ani 100 Mbps.

Skutečností je, že standardní architektura Intel PC, kterou pro náš firewall Kernun používáme, nezvládne gigabitovou rychlostí komunikovat ani jako klient nebo server, natož aby fungovala jako router nebo dokonce firewall. Na výkonných sestavách máme naměřeno přes 300 Mbps (včetně funkce aplikačního proxy firewallu). Zkoumáme možnosti sběrnice PCI-X s frekvencí 133 MHz, výsledky měření nejsou v době psaní tohoto textu známy.

Mnohokrát jsme byli svědky rozhodování o pořízení firewallu, v němž hlavním kritériem byla rychlost. Nic proti rychlosti, ale výsledkem byl prakticky vždy nákup gigabitového routeru, který také „trochu firewalluje“. Výrobci se tomuto trendu podřizují, a ve snaze zalíbit se zákazníkovi prezentují jako firewall i to, co si takové označení nezaslouží.

Podobnou koncepční chybou bývá začasté zanedbání okolních prvků v síti. Drahý a ostře sledovaný firewall je zapojen do sítě prostřednictvím nejlevnějšího zařízení na trhu. Zaznamenali jsme i případy, kdy byla použita konfigurace dvou firewallů v clusteru (druhý přebere automaticky provoz v případě, že detekuje výpadek prvního), přičemž oba byly zapojeny do stejného rozbočovače (hubu), vhodného sotva pro domácí použití.

2 Uživatelské rozhraní aneb Stíhačka na tlačítko

Snad nejčastější stížnost na náš firewall Kernun, kterou slýcháme, je chybějící grafické uživatelské rozhraní. Už téměř čtyři roky vzdorujeme a v nejnovější verzi máme stále jen textové rozhraní, přestože vylepšené o kontextovou nápovědu a automatické doplňování příkazů.

Jednoduché „klikací“ rozhraní vede k jednoduchým řešením složitých problémů a k myšlenkovým zkratům. Moderní aplikační proxy firewall představuje složitý systém, a jako takový se nedá ovládat několika málo tlačítky. Stíhací letadlo má také mnoho ovládacích prvků (dle rozšířené

logiky by se možná mělo spouštět i zastavovat stisknutím jediného tlačítka *Start*).

Špatně nakonfigurovaný firewall je zpravidla horší než žádný firewall. Falešný pocit bezpečí ohrožuje zejména ty sítě, jejichž administrátoři nevědí zcela přesně, co činí. Nulová koncepce, potřeba rychle provést změny (nejlépe včera) vede ke stresu a nutí nepřemýšlet nad podstatou problému. To vše vídáme denně, to vše je nebezpečné a znamená zvýšení celkového rizika.

3 Bezpečnostní politika aneb 2 kg bezpečnosti

Stejně jako nutnost nasazení firewallu se v posledním desetiletí dostala do povědomí prakticky všech odpovědných osob, stává se nyní podobně populární bezpečnostní politika informačních systémů. Přestože pořadí mělo být opačné, nic naplat, buďme vděčni alespoň za to.

V rozpočtové sféře je významným hybatelem zákon č. 365/2000 Sb. o informačních systémech veřejné správy. Jenže každá mince má dvě strany, a v tomto případě jsme svědky toho, jak se bezpečnostní politiky sestavují na poslední chvíli jen proto, aby bylo učiněno zadost formalitám zákona.

Výsledné bezpečnostní politiky nestojí za moc, a rozhodně se podle nich nedá nakonfigurovat firewall. Účelná bezpečnostní politika, jejíž vypracování nabízíme, je totiž nákladná. Navíc není stabilní, se změnami ve fungování organizace musí být neustále aktualizována. A v některých komerčních firmách dochází k zásadním změnám tak často, že na vytvoření skutečné bezpečnostní politiky v oblasti informačních technologií prostě rezignovali.

Obě tyto skutečnosti reprezentuje asi nejlépe teze: Hlavně ať to funguje. Není důležitá *bezpečná* konfigurace firewallu, ale *funkční*. Hotovou a fungující konfiguraci sítě si pak zákazník přeje doplnit „trochou té bezpečnosti“ (a my obvykle přibalíme o 11 deka víc).

Důsledkem je následující, značně typický scénář: Zákazník se pouští do implementace nové komponenty svého informačního systému. Dodavatel mu nadiktuje požadavky, které zákazník zcela nekriticky přijímá. Mezi nimi je i „otevření portu 12345“, tečka. V této surové podobě dostaneme zadání, které často neupřesní ani rozhovor se zmíněným dodavatelem. Vysvětlíme sice zákazníkovi, jakým rizikům se může vystavit, ale dostáváme očekávanou odpověď:

Hlavně to musí fungovat! Pak si teprve budeme moci dovolit uvažovat o bezpečnosti.

Mnohdy se přístup správců mění po nasazení firewallu do provozu. Díky velmi podrobnému logování si mohou prohlížet statistiky a konfrontovat svou představu o využívání linky se skutečností[1]. Podnětem může být sice pouhá zvědavost, výsledkem však v příznivějších případech bývá definice pravidel používání Internetu a mnohem přísnější (a tudíž bezpečnější) konfigurace firewallu.

Do stejné kategorie řadíme mnoho dalších neduhů, z nichž jeden nás jako producenta firewallu pálí nejvíce: Nedodržování standardů není v případě velkých softwarových firem považováno za jejich chybu; pokud je taková skutečnost příčinou dysfunkce firewallu, jsme to my, na koho se zákazník obrátí. Lze si jen povzdechnout: Pohnout horou holýma rukama nelze a bojovat s větrnými mlýny k cíli nevede.

4 Neznámý firewall aneb Ať žijí duchové

Firewall je jednou z nejtajemnějších a nejzáhadnějších věcí na světě. Vždyť kdykoli se vyskytne nějaký problém v síti, je firewall jako první v podezření. Nedokážu posoudit, jestli se tak děje spravedlivě na základě nějakého statistického pozorování.

Za mnohé z těchto problémů přitom může absence jedné důležité fáze budování komplexního informačního systému, a tou je integrace. Jednotlivé komponenty, subsystémy a komunikační prvky se vrší na sebe, ztrácí se přehled a řád. V takovém prostředí pak snadno změna v jedné části systému ovlivní funkci jiné části, aniž by to bylo na první pohled patrné. A už je tady telefonát, e-mail nebo ticket systémové podpory, už naši technici suplují činnost integrátora a zjišťují, jak se vlastně jednotlivé subsystémy chovají při síťové komunikaci.

5 Zero time exploit aneb Čekání na další chybu

Další chyba ve Windows, ještě že máme firewall!

Tak uvažuje spousta správců sítí, tedy těch osvícenějších, kteří bezpečnostní problémy alespoň sledují. Technické detaily útoků jsou však

[1] Někdy s kolegy (jen napůl žertem) říkáme, že vlastně pracujeme v pornografickém průmyslu.

komplikované. Zjistit, zda a jak je ohrožena konkrétní síť, není jednoduché. Je zřejmé, že firewall dokáže mnohem snadněji zabránit *aktivním* útokům (útočník iniciuje zlovolnou komunikaci) než těm *pasivním* (zlovolný kód čeká na aktivaci nic netušícím klientem).

Právě ta druhá kategorie by neměla nechat správce sítě chladným. Přestože firewall může mít nástroje, s jejichž pomocí lze útoku zabránit, je většinou nutné tyto nástroje explicitně pro každý nový útok nastavit (např. zakázat jistý vzorek URL).

Tato povaha obrany, vyžadující jistou reakci, je v dnešním zrychleném světě informací značnou nevýhodou. Časové okno mezi veřejným oznámením bezpečnostní chyby a rozšířením běžně dostupného kódu, který chybu zneužívá, se zmenšuje (právě kód zneužívající chybu a dostupný ve stejné chvíli jako informace o chybě se nazývá *zero time exploit*).

Problematický obsah může být stažen klientem mnoha různými způsoby. V moderním firewallu je nutné uplatňovat prakticky shodná omezení uvnitř HTTP proxy, FTP proxy, SMTP proxy, POP3 proxy a dalších. Vždyť není nic těžkého umístit zlovolný kód na FTP server místo na Web, a příslušné odkazy ještě navíc rozeslat e-mailem. Sebemenší opomenutí v konfiguraci, jakýkoli otevřený způsob komunikace bez kontroly obsahu pak znamená obejití všech tolik složitých mechanismů a nastavení.

6 Nedůvěra k českému výrobci aneb Made in USA

Jak může česká firma vytvořit kvalitní firewall? Těm americkým určitě nesahá ani po kotníky!

S takovým postojem se setkáváme neustále. Přitom soupeříme s produkty, o kterých dobře víme, že jsou v mnohém ohledu horší. Často je to zahraniční vedení firmy, které diktuje výrobce, jehož software se bude nakupovat.

Jsme přesvědčení, že podstatnějším kritériem je dostupnost podpory a odborné pomoci. Ze zkušenosti dobře vím, že podpora u většiny zámořských softwarových produktů je prakticky nulová. Jistě, existují u nás certifikovaní odborníci, partneři apod., hluboká znalost jim přesto nezřídka chybí. Zákazník se určitě nesetká s takovou pružností a ochotou přizpůsobit produkt jeho potřebám.

Mírný obrat nastává spolu s uváděním zákona č. 365/2000 Sb. o informačních systémech veřejné správy do praxe; týká se ovšem, jak jinak,

jen rozpočtové sféry. Jak jsme si sami ověřili, výhodu mají produkty dle tohoto zákona atestované, což podstupují snadněji právě domácí výrobci.

7 Závěr

Co uvést na závěr? Samozřejmě, není to zase tak zlé. Popsané neduhy se sice vyskytují velmi často, je ale třeba přiznat, že zákazníci nám z velké většiny naslouchají a nechají si poradit. Na poli bezpečnostních politik i integrace se situace mírně zlepšuje.

To vše je příslibem pro budoucnost bezpečnosti podnikových sítí. Nemohu se však zbavit pocitu, že zdaleka nejvlivnější faktor mají v rukou mamutí softwaroví producenti a jejich nekritičtí konzumenti.

Hardwarové bezpečnostní moduly – API a útoky

Jan Krhovják, Daniel Cvrček[1], Vašek Matyáš[2]

E-mail: xkrhovj@informatics.muni.cz,
Daniel.Cvrcek@cl.cam.ac.uk, matyas@informatics.muni.cz

Abstrakt

Příspěvek pojednává o vybraných aspektech hardwarových bezpečnostních modulů a zaměřuje se na útoky na a přes aplikační programovací rozhraní (API). Nejdříve se kromě architektury kryptografických modulů seznámíme se základními požadavky na jejich bezpečnost a s americkými normami FIPS 140-1 a FIPS 140-2, jimiž jsou tyto požadavky obvykle specifikovány. Dále se pak budeme věnovat samotné problematice bezpečnosti API, a také popisu útoků, které jsou často umožněny chybným návrhem mnohých běžně používaných aplikačních programovacích rozhraní.

1 Úvod

Zajištění bezpečné komunikace především u distribuovaných systémů a aplikací vyžaduje použití vhodných mechanizmů a prostředků pro zajištění integrity jejich kryptografických funkcí a důvěrnosti příslušných šifrovacích klíčů. U běžně používaných výpočetních systémů toho lze dosáhnout jen velmi obtížně, protože jejich hardware je typicky zcela fyzicky nezabezpečen a software obsahuje velké množství chyb. To bylo hlavním důvodem návrhu a vytvoření hardwarových bezpečnostních zařízení (HSM – Hardware Security Module), do jejichž fyzicky zabezpečeného prostředí se přesunulo provádění veškerých kryptografických operací.

[1] Práce na příspěvku byly uskutečněny především během působení v Computer Laboratory, University of Cambridge, UK.

[2] Práce na příspěvku byly také prováděny během předchozího pobytu na University College Dublin a nynějšího u Microsoft Research Ltd. (Cambridge, UK).

Oblastí, jež odstartovala vývoj hardwarových bezpečnostních zařízení, bylo bankovnictví. A hlavní skupinou aplikací pak elektronické transakce v bankovních sítích (např. VISA, MasterCard, American Express) zahrnující komunikaci jednotlivých bank a komunikaci s jejich peněžními bankomaty (ATM). Tyto rozsáhlé ATM sítě[3], do kterých je v dnešní době sdružováno stále více bank, umožňují zákazníkům provádět finanční transakce téměř z kteréhokoliv místa na světě a bankám také samozřejmě přinášejí větší obrat a zisky. Jednotlivé banky ani zákazník si však nemohou vzájemně důvěřovat a úlohou HSM je, kromě zabezpečení důvěrného přenosu dat, také bezpečná správa kryptografických klíčů a jiných citlivých dat (např. PINů) tak, aby se předešlo podvodům ze strany klientů i zaměstnanců bank.

Hardwarová bezpečnostní zařízení poskytují bezpečné prostředí pro provádění citlivých operací, ke kterým je možné přistupovat za použití přesně definovaného softwarového rozhraní – tzv. aplikačního programovacího rozhraní (API). To by mělo být navrženo tak, aby nemohlo dojít ke zneužití či kompromitování chráněných dat uložených v HSM. Snaha o co nejflexibilnější návrh těchto zařízení, a podpora mnoha standardů a norem, však činí jednotlivá API příliš složitá a jejich správnou funkčnost lze pak jen stěží zaručit. Důkazem toho je stále narůstající počet útoků na API různých HSM, jejichž analýzou se budeme v tomto příspěvku zabývat.

1.1 Základní terminologie a pohled na bezpečnost

Hardwarová bezpečnostní zařízení, někdy též označovaná jako *kryptografické koprocesory* či *kryptografické moduly*, jsou bezpečnostní zařízení určená k bezpečnému provádění kryptografických operací v jinak nedůvěryhodném prostředí. Lze mezi ně zařadit také *čipové karty*, které se používají k autentizaci nebo k uchovávání citlivých dat, či *kryptografické akcelerátory*, které slouží k urychlování kryptografických operací.

Koprocesory i akcelerátory bývají většinou nainstalovány v tzv. *hostitelských zařízeních*, což mohou být například osobní počítače, velké výpočetní servery nebo specializované bankovní systémy. Z bezpečnostního hlediska může být hostitelské zařízení *nedůvěryhodným prostředím*,

[3] V tomto příspěvku bude pojem *ATM síť* značit vždy síť peněžních bankomatů, a význam zkratky ATM je tedy nutno interpretovat nikoliv jako Asynchronous Transfer Mode, ale jako Automatic Teller Machine.

zvlášť pokud je umístěno na veřejně přístupném místě a je bez jakékoliv fyzické ochrany.

Převážná část příspěvku popisuje útoky na HSM. Jako *útok* označujeme postup, pomocí něhož získáme data, která mají být pro útočníka nepřístupná, nebo postup, kterým provede operaci, ke které není autorizován.

1.2 Architektura kryptografických modulů

Architektura kryptografických modulů je do značné míry závislá na jejich typu. Zakladní architektura vychází z klasické von Neumannovy architektury. Oproti běžným počítačům je ovšem množina základních bloků rozšířena o mechanizmy fyzické ochrany (odolnost proti fyzickým útokům), generátory náhodných čísel či speciální koprocesory pro urychlení specifických kryptografických operací. Na druhou stranu neobsahují např. běžné V/V obvody, což značně snižuje složitost operačního prostředí. Je zřejmé, že odolnost proti průnikům (fyzickým útokům) nebude u čipových karet nikdy realizována stejným způsobem jako u kryptografických koprocesorů či bankomatů. Jestliže ovšem porovnáme funkční schémata, základní bloky jsou stejné.

Jádrem celého zařízení je procesor, který řídí veškeré vstupní a výstupní operace, zpracovává přerušení a stará se o správu paměti. Navíc úzce spolupracuje s případnými dalšími procesory sloužícími k provádění specializovaných kryptografických operací.

Paměť určená pro uchování citlivých informací je většinou napájena přídavnými bateriemi a obsahuje tajné klíče, které mohou být v případě detekce průniku vymazány. Generátor náhodných čísel je obvyklou součástí kryptografických modulů, protože kryptograficky bezpečný zdroj náhodných dat je pro tato zařízení naprostou nutností.

Na kryptografické čipové karty (smart cards) lze pohlížet jako na levné kryptografické moduly s menší výpočetní silou.

2 Bezpečnostní požadavky na kryptografické moduly

Bezpečnostní požadavky na kryptografické moduly jsou specifikovány v normě FIPS[4] 140-2 [11], která od 25. listopadu 2001 zcela nahradila starší normu FIPS 140-1 [10] z roku 1994. Norma definuje čtyři

[4] Federal Information Processing Standard.

úrovně zabezpečení, které pokrývají široké spektrum možností potenciálního použití kryptografického modulu v různých aplikacích a prostředích. Jednotlivé úrovně jsou specifikovány požadavky v jedenácti oblastech (viz 2.1) vztahujících se k bezpečnému návrhu a implementaci modulu. Vyšší úrovni odpovídají i vyšší požadavky na zabezpečení.

Úroveň 1 – definuje nejnižší stupeň ochrany. Specifikovány jsou pouze základní bezpečnostní požadavky, jako je např. používání schválených algoritmů. Nejsou vyžadovány žádné specifické mechanizmy fyzické ochrany ani speciálně ohodnocený operační systém. Typickými příklady zařízení spadajících do této úrovně jsou osobní počítače.

Úroveň 2 – rozšiřuje fyzické zabezpečení modulu tím, že vyžaduje zajištění evidence průniků, která bývá nejčastěji realizována použitím kvalitních zámků či pečetí. Nutností je také autentizace založená na *rolích*[5]. Tato úroveň povoluje spouštění softwarových a firmwarových částí kryptografického modulu na obecném počítačovém systému pouze s operačním systémem[6], který splňuje alespoň úroveň EAL2 ohodnocenou podle CC[7]. Typickými příklady zařízení spadajících do této úrovně jsou čipové karty.

Úroveň 3 – dále rozšiřuje požadavky na fyzické zabezpečení. Přístup k citlivým informacím udržovaným uvnitř modulu již není pouze pasivně chráněn, např. za použití silných ochranných krytů, ale musí existovat mechanizmy, které v případě detekce průniku citlivá data vymažou. Požadována je i autentizace založená na ověřování identity, což je rozšíření autentizace založené na rolích. Citlivá data, která opouštějí modul v nezašifrované podobě, by měla používat speciálních fyzicky oddělených portů či rozhraní, s jejichž pomocí lze vytvořit důvěryhodný kanál. Softwarové a firmwarové části modulu mohou být spouštěny pouze operačním systémem, který splňuje alespoň úroveň EAL3 ohodnocenou podle CC. Příkla-

[5]Při tomto typu autentizace nemusí být ověřena identita operátora.

[6]Tímto je míněn operační systém uvnitř hardwarového modulu, nikoliv na hostitelském zařízení.

[7]Common Criteria – mezinárodní kritéria ohodnocení IT bezpečnosti. Standard FIPS se opírá o CC všude tam, kde je vyžadována validace funkčních vlastností.

dem zařízení splňujícího FIPS 140-1 a spadajícího do této úrovně je Luna CA.

Úroveň 4 – definuje nejvyšší stupeň zabezpečení. Zařízení na této úrovni bývají určena k provozování v zcela nechráněných prostředích, a proto by měla být schopna detekovat a reagovat na všechny známé neautorizované pokusy o fyzický průnik. Navíc je také požadována specifikace vnějších provozních podmínek modulu, které musí být za provozu dodrženy (např. povolený rozsah napětí či teplot). Důvodem je existence útoků, které využívají překročení provozních podmínek k získání citlivých informací. Kryptografický modul tedy musí buď obsahovat senzory, s jejichž pomocí je možné vnější podmínky kontrolovat a případně citlivá data smazat, nebo provedení série přísných testů, jež prokážou, že je schopen chránit citlivá data i při práci mimo rozsah jeho operačních hodnot. Softwarové a firmwarové části modulu mohou být spouštěny pouze operačním systémem, který splňuje alespoň úroveň EAL4 ohodnocenou podle CC. Příkladem zařízení splňujícího FIPS 140-1 a spadajícího do této úrovně je IBM 4758.

2.1 Oblasti bezpečnostních požadavků

V této části jsou v jednotlivých oblastech specifikovány bezpečnostní požadavky, vůči kterým je kryptografický modul nezávisle testován. Po ukončení testů získá modul celkové ohodnocení tak, že jeho výsledná úroveň zabezpečení je rovna minimální úrovni dosažené i třeba jen v jediném testu. V mnoha oblastech klade standard požadavky i na obsah povinné, prodejcem poskytované dokumentace. Následující odstavce popisují rozsah požadavků kladených na jednotlivé aspekty bezpečnosti.

Dokumentace kryptografického modulu – by měla specifikovat hardwarové, firmwarové a softwarové komponenty kryptografického modulu, kryptografickou hranici modulu a fyzickou ochranu modulu. Je třeba specifikovat fyzické porty, logická rozhraní a všechny definované datové vstupy a výstupy. Dokumentace by měla také obsahovat výčet všech bezpečnostních funkcí modulu včetně všech operačních modů a specifikovány by měly být i veškeré k bezpečnosti se vztahující informace (např. kryptografické klíče, autentizační data) a bezpečnostní politika modulu.

Porty a rozhraní – kryptografický modul by měl omezovat veškerý datový tok a fyzický přístup k logickým rozhraním, která definují vstupní

a výstupní body modulu. Rozhraní musí být alespoň logicky odděleny, ale mohou sdílet společný fyzický port. Norma specifikuje čtyři logická rozhraní pro vstup a výstup. První je určené pro vstup příkazů, signálů a kontrolních dat. Druhé pro výstup stavových dat, signálů či fyzických indikátorů stavu (tj. LED diody a displeje). Zbylá dvě pak slouží pro vstup a výstup všech dat (včetně kryptografických klíčů či nezašifrovaných dat). Za speciální vstup do kryptografického modulu je považován i přívod napájení, či zdroj hodinového kmitočtu.

Role, služby a autentizace – kryptografický modul by měl pro operátory[8] podporovat autorizované *role*, s přiřazenými službami. Pokud jsou podporovány paralelní sezení, musí být logicky zcela odděleny (procesorový čas, paměť) operace prováděné v jednotlivých sezeních. Jeden operátor může mít přiděleno i více rolí, z nichž následující tři by měly být vždy modulem podporovány:

1. Uživatelská role – umožňuje přístup k bezpečnostním službám, a provádění kryptografických funkcí či jiných operací.

2. Role bezpečnostního úředníka – umožňuje inicializaci a konfiguraci zařízení (např. vkládání či změnu kryptografických klíčů a funkce auditu).

3. Role pro údržbu – je určena pro služby fyzické a logické údržby (např. diagnostika hardware či software). Při použití této role jsou veškeré citlivé informace bez ochrany automaticky vymazány. Tato role ale nemusí existovat, pokud není podporována údržba zařízení.

Pojem *služby* pokrývá všechny operace a funkce, které mohou být kryptografickým modulem prováděny. Operátor by měl mít vždy k dispozici služby poskytující informace o stavu zařízení a provádějící testování modulu a alespoň jednu schválenou bezpečnostní funkci. Dále může být také požadována *autentizace* operátora, jejíž pomocí modul ověří, zda je operátor autorizován k osvojení požadované role a k používání služeb s ní spjatých. Podporována by měla být alespoň jedna z následujících metod:

1. Autentizace založená na rolích – modul vyžaduje, aby si operátor vybral jednu či více rolí a autentizuje její (resp. jejich) převzetí. Identita samotného operátora není testována.

[8] Jedinec či proces, který má přístup ke kryptografickému modulu a jeho službám.

2. Autentizace založená na identitách – oproti předchozímu případu je navíc testována i identita operátora.

Při implementaci těchto mechanizmů se jako autentizační data využívají například hesla, klíče, PINy, tokeny či biometriky (kdy náhodný pokus o přístup má šanci úspěchu menší než $1 : 10^6$). Tato data by měla být uvnitř modulu chráněna proti modifikaci, záměně či odhalení. Po restartu zařízení je vždy nutná nová autentizace.

Konečně stavový model – každý modul (na úrovni 4) by měl být specifikován pomocí konečně stavového přechodového diagramu, který obsahuje všechny operační a chybové stavy modulu, přechody mezi těmito stavy a události, které přechody způsobují nebo jsou jimi způsobeny.

Fyzická bezpečnost – tato oblast definuje požadavky na fyzické bezpečnostní mechanizmy kryptografického modulu, které jsou zvlášť specifikovány pro tři základní typy modulů:

1. Jednočipové – tyto moduly obsahují jeden integrovaný obvod a nemusí být nijak fyzicky chráněny (předpokládá se ochrana hostitelským zařízením). Typickým příkladem jsou čipové karty.

2. Vícečipové vestavěné (embedded) – obsahují jeden a více integrovaných obvodů, které však také nemusí být fyzicky chráněny krytem. Příkladem jsou rozšiřující karty.

3. Vícečipové autonomní – integrované obvody jsou v tomto případě již zcela fyzicky chráněny svým krytem. Tyto moduly se používají v nechráněném a v nedůvěryhodném prostředí. Příkladem jsou kryptografické routery.

V závislosti na typu modulu jsou pak stanoveny požadavky pro jednotlivé úrovně zabezpečení.

Operační prostředí – má smysl pouze u zařízení, která umožňují přihrávání a spouštění nových částí programového vybavení. Podstatnou součástí prostředí je operační systém, ten je rozdělen na modifikovatelnou (např. firmware v RAM) a nemodifikovatelnou (např. firmware v ROM) část.

Správa klíčů – tato oblast definuje požadavky na celý životní cyklus kryptografických klíčů: generování, ustanovení, verifikaci, distribuci, uložení či vymazání. Její součástí je také specifikace generátoru náhodných

čísel, jehož pomocí mohou být tajné klíče vytvářeny. Pro všechny úrovně zabezpečení je vyžadováno, aby tajné či soukromé klíče byly kryptografickým modulem chráněny před neautorizovaným čtením, modifikací či záměnou.

Elektromagnetické interference a kompatibilita – v této sekci jsou definovány požadavky na elektromagnetické interference a kompatibilitu pro FCC[9].

Auto-testy – zde jsou definovány požadavky na testy, které musí být provedeny modulem při svém spuštění či restartu, a které ověřují jeho správnou funkčnost a integritu. Proběhne-li testování neúspěšně, musí modul vstoupit do chybového stavu a chybu ohlásit. V tomto stavu nelze provádět žádné kryptografické operace. Kromě testování integrity software/firmware jsou testovány například i kryptografické algoritmy, generátor náhodných čísel či korektnost soukromého a veřejného klíče.

Záruka návrhu – tato oblast se týká použití nejlepších technik a postupů pro vývoj, správu konfigurace, nasazení a používání. Poskytuje také záruky řádného testování, doručení a instalace. Důraz je kladen i na uživatelskou dokumentaci.

Zmírnění jiných útoků – tato část pojednává o zmírnění dalších útoků, na něž může být kryptografický modul náchylný. Jedná se především o známé útoky, pro něž v době vydání tohoto standardu nebyly ještě ustanoveny testovatelné požadavky (např. časová analýza, výkonová analýza), nebo které převyšují rámec této normy (např. program TEMPEST).

2.2 Srovnání norem FIPS 140-1 a 140-2

FIPS 140-2 je relativně nový standard, a proto je mnoho v současné době používaných modulů certifikováno pouze podle FIPS 140-1. Obě normy mají podobnou strukturu, avšak k drobným změnám či upřesněním došlo ve všech částech. Jejich podrobné srovnání lze nalézt v NIST SP[10] 800-29 [12]. Norma FIPS 140-2 je v některých částech jinak strukturována a došlo také k celkovému upřesnění a sjednocení použité terminologie. Asi nejpatrnějšími změnami oproti FIPS 140-1 je přidání nové části týkající se zmírnění jiných útoků a fyzická/logická separace portů. K dalším

[9] Federal Communications Commission – americká státní agentura zodpovědná za regulaci federálních komunikací.

[10] National Institute of Standards and Technology Special Publication.

změnám patří například zesílení požadavků na autentizační mechanizmy a na testování modulu. V důsledku vzniku nových norem je již také operační systém ohodnocován podle CC (namísto TCSEC[11]).

3 Popis útoků na a přes API

Cílem této části je podrobný rozbor útoků na a přes API, jejichž význam spočívá především v demonstraci nedostatků a chybného návrhu API současných kryptografických modulů. Ukazuje se, že postupný vývoj těchto produktů a snaha o univerzálnost návrhu těchto zařízení, a tudíž i podpora mnoha standardů či norem, vyústila v až příliš rozsáhlá API, jejichž správnou funkčnost lze jen stěží zaručit. Výzkum týkající se bezpečnosti API odstartoval Ross Anderson v [1].

3.1 Úvod do problematiky API pro HSM

API tvoří jediné komunikační rozhraní mezi kryptografickým koprocesorem a vnější aplikací. Jejich pomocí je realizován přístup k veškerým operacím, které koprocesor provádí. Na základě funkcí API jsou budovány protokoly, které bývají typicky tvořeny posloupností tří až pěti zpráv, které si předávají (resp. přeposílají) jednotlivé strany protokolu.

Návrh protokolů je již sám o sobě velmi obtížnou záležitostí, a to i přes malý počet vyměňovaných zpráv, které mohou být útočníkem zmanipulovány (HSM nepodporují sezení, což znamená, že každá zpráva musí být samopopisná). API kryptografického koprocesoru obsahuje typicky 30–500 funkcí s mnoha parametry, což poskytuje dostatečně velmi velký prostor k chybám a vzniku útoků. Výsledkem útoků, které byly objeveny, je vždy obejití definované bezpečnostní politiky zařízení.

V tomto příspěvku se budeme zabývat výhradně kryptografickými API. Ty lze dále rozdělit na standardní a finanční kryptografická API[12]. Standardní kryptografická API poskytují pouze základní funkcionalitu, která je požadována po bezpečnostním zařízení (např. správa klíčů či šifrování). Oproti tomu finanční kryptografická API poskytují navíc funkce pro specifické finanční operace (např. manipulace s PINy či podpora SET), které jsou vyžadovány bankovním sektorem. Mezi běžně komerčně používané API patří například The Common Cryptographic Architecture

[11] Trusted Computer System Evaluation Criteria – někdy též označovány jako *Oranžová kniha*.

[12] V praxi jsou však většinou součástí pouze jednoho společného API.

(CCA) od IBM, The Compaq-Atalla API, Public Key Cryptography Standard (PKCS) #11 od RSA nebo The Thales-Zaxus-Racal API.

Abychom vytvořili alespoň trochu přehledný seznam existujících útoků, rozdělili jsme je do skupin podle typů chyb, kterých využívají. Těmi základními oblastmi jsou klíče a jejich integrita, nedostatečná kontrola parametrů funkcí, nevynucování bezpečnostní politiky.

3.2 Klíče a jejich integrita

Problém evoluce HSM a snaha o dodržení zpětné kompatibility je nejpatrnější u pokračujícího používání slabého kryptografického algoritmu DES. Ačkoliv nové moduly už dokáží pracovat např. s algoritmem 3DES (se dvěma klíči), tak bylo nalezeno velké množství útoků, které využívají nedostatečné ochrany před změnou typu klíče z 3DES na DES a relativně snadného útoku na slabší algoritmus.

3.2.1 The Meet in the Middle Attack

Malá velikost šifrovacích klíčů DES umožňuje využít špatný návrh rozlišování typů klíčů a neexistenci omezení na generování klíčů (např. limity omezující počet vygenerovaných klíčů). Tento fakt umožňuje snížit výpočetní náročnost hledání DES klíče. Mnoho kryptografických koprocesorů dokáže vygenerovat desetitisíce klíčů řádově během minut. Jestliže útočník vygeneruje 2^{16} klíčů, sníží tak výpočetní náročnost nalezení alespoň jednoho klíče z původních 2^{55} na 2^{39}. Prohledání takto redukovaného prostoru klíčů zabere i na domácím PC pouze několik dnů, s použitím speciálního zařízení využívající FPGA technologii několik hodin.

Myšlenka celého útoku je rozdělit výpočetní složitost na výpočetní a paměťovou složitost. Nejprve se každým z vygenerovaných 2^{16} klíčů zašifruje stejný testovací vzorek a výsledek se uloží. Poté se systematicky prohledává klíčový prostor a stejný testovací vzorek se každým klíčem zašifruje a porovná se všemi uloženými vzorky. Dojde-li při porovnávání ke shodě, znamená to, že byla nalezena hodnota jednoho tajného klíče.

Je-li tento klíč oprávněn k šifrování dalších klíčů uložených v HSM (tzv. terminální klíč), lze jím přešifrovat veškerá jinými terminálními klíči chráněná data a klíče, které pak útočník snadno dešifruje. Tímto způsobem je možné kompromitovat osm z devíti typů klíčů, které používá Visa Security Module (VSM). Podrobně je tento útok popsán v [4].

Velmi pěknou variantu útoku je možno aplikovat i na kryptografický modul Prism. Při vynaložení stejného úsilí jako v předchozím případě lze získat dokonce hlavní klíč celého zařízení. V tomto kryptografickém modulu se hlavní (DES) klíč ustanovuje manuálně z jednotlivých částí, které jsou v modulu postupně XORovány (operace $\oplus$). Po vložení každé části klíče je vrácen kontrolní vektor (pro ověření korektního nahrání nové části klíče) zašifrovaný nově vytvořeným klíčem. Chybou v API tohoto zařízení je, že jakýkoliv uživatel může pokračovat v přidávání nových částí hlavního klíče, a tím i v získávání dalších variant zašifrovaného kontrolního vektoru. Jestliže tak vytvoří 2^{16} variant zašifrovaného kontrolního vektoru postupuje dále analogicky s předchozím případem.

3.2.2 Conjuring Keys From Nowhere

Kouzlení klíčů je považováno za útok[13] umožňující neautorizované generování klíčů ukládaných mimo kryptografický koprocesor. V podstatě se jedná o náhodné vytvoření zašifrovaného klíče, který se podstrčí koprocesoru. Po dešifrování je hodnota klíče také náhodná a v případě DES má s pravděpodobností $1/2^8$ správnou paritu[14]. V případě 3DES má správnou paritu s pravděpodobností $1/2^{16}$, což je stále dosažitelné. Tento útok lze aplikovat i na mnohé současné kryptografické moduly (např. IBM 4758 s CCA API), které používají formáty klíčů bez jakéhokoliv doplnění.

Na první pohled toto nevypadá přímo jako útok, protože útočník hodnotu odšifrovaného klíče nezná a není schopen dešifrovat data tímto klíčem zašifrovaná, ale takto vložené klíče mohou být využity ke složitějším útokům na API. Obrana spočívá v pečlivějším návrhu formátu klíčů obsahujícím větší množství entropie (např. před jeho zašifrováním přidat kontrolní součet a časové razítko).

3.2.3 3DES Key Binding Attack

Tento útok popsaný v [2, 4] je důsledkem nedostatečné vazby jednotlivých částí 3DES klíčů a jeho výsledkem je kompromitování většiny ko-

[13] I přesto, že některé starší moduly používaly tuto techniku k regulérnímu generování klíčů.

[14] DES používá lichou paritu a jako paritní je považován nejméně významný bit každého oktetu.

procesorem chráněných klíčů. CCA sice rozlišuje[15] mezi levou a pravou částí klíče, ale již nespecifikuje příslušnost ke konkrétnímu klíči. Tím je umožněna nekorektní manipulace s jeho jednotlivými polovinami. CCA vyžaduje u některých typů klíčů, aby byly nejen typu 3DES, ale aby se také obě 64 bitové části klíče lišily. Pro generování klíčů ovšem druhá podmínka není vyžadována (z důvodu zpětné kompatibility).

Útočníkovi tedy tímto útokem stačí vygenerovat velké množství klíčů se stejnými polovinami (replicated keys) a stejného typu jako požadovaný klíč, pomocí *Meet in the Middle* útoku nalézt hodnoty dvou z těchto klíčů (prohledávání 2^{41} možností) a výměnou jejich polovin vytvořit dva 3DES klíče s odlišnými polovinami. Je-li nalezeným klíčem exportní klíč, můžeme nyní s jeho pomocí exportovat a poté dešifrovat všechny klíče v HSM určené k exportu.

Získat však lze i klíč, který nemá povolen export. Stačí jen zaměnit jednu jeho polovinu s polovinou známého klíče. Tím vzniknou dva klíče, jejichž jedna polovina je známá a druhou získáme prohledáváním prostoru 2^{56} (hledáme oba klíče současně). To je již samozřejmě práce pro speciální hardware či distribuované systémy. Pokud je uplatňována přísnější kontrola přístupu a útočník nemá povoleno vytvářet klíče se stejnými polovinami, musí vygenerovat standardní DES klíč a změnit obsah jeho kontrolního vektoru (struktura obsahující informace o klíči) na LEFT HALF OF A 3DES KEY. K tomu je možné použít útok na import klíčů [2, 3, 4].

3.3 Nedostatečná kontrola parametrů funkcí – PINy

U bankovních HSM je jednou z nejcitlivějších oblastí práce s PINy bankovních karet. Ověřování PINů bylo ovšem hlavní motivací pro zavádění HSM v bankovnictví, protože umožňovaly klientům bank používat své bankovní karty kdekoliv po světě. Je pozoruhodné, jak špatně jsou chráněny citlivé parametry právě těch funkcí zajišťujících operace s PINy. PINy jsou formátovány do tzv. PIN-bloků (CPB), což jsou 8 bajtové struktury. Ty se po zašifrování nazývají EPB (encrypted PIN block).

Útoky k získání PINů – *PIN Recovery Attacks* tvoří snad největší třídu útoků na současné HSM. Tyto útoky demonstrují techniky, jejichž pomocí lze ze zašifrovaného PIN-bloku bez znalosti klíče získat hodnotu PINu. K jejich provedení mnohdy stačí pouze přístup ke kryptografic-

[15] Pro zjednodušení uvažujme pouze dvouklíčový 3DES.

kému finančnímu API daného zařízení (s řádně nainstalovanými klíči) a jeden zašifrovaný PIN-blok (EPB).

Kromě toho, že jsou útoky extrémně rychlé a snadno implementovatelné, lze je většinou aplikovat i na více druhů běžně používaných API[16], čímž postihnou mnohem více HSM. Dále v této části vycházíme především z [5, 7, 9].

3.3.1 Decimalisation Table Attacks

Typická verifikační funkce na základě validačních dat (obvykle je to číslo účtu) vygeneruje PIN a porovná jej s PINem získaným z EPB[17]. Bezpečnostním problémem těchto funkcí jsou právě metody generování PINů vycházející ze starých metod používaných bankomatem IBM 3624. Jedním z parametrů těchto funkcí je decimalizační tabulka umožňující převod čísel ze šestnáctkové na desítkovou soustavu. S touto tabulkou lze ovšem dělat zajímavé věci. Podívejme se ale nejdříve, jak se vlastně PIN generuje.

Techniky generování a verifikace PINů

Existuje mnoho používaných metod generování a verifikace PINů, jejichž typickými příklady jsou metody IBM 3624 a IBM 3624 Offset. IBM 3624 generování PINů je založeno na validačních datech (např. číslo účtu – PAN), která jsou zašifrována PIN generujícím klíčem a požadovaný počet číslic je převeden do desítkové soustavy (decimalizován) a zvolen jako PIN.

Verifikace pak probíhá analogicky, avšak PIN generující klíč se nazývá PIN verifikující klíč a vypočítaný PIN je nakonec porovnán s PINem získaným z EPB. Metoda IBM 3624 Offset navíc použitím offsetů umožňuje změnu PINu zákazníkem. Generování zde probíhá stejně jako v předchozím případě, ale výsledek se nazývá IPIN (intermediate PIN) a offset je získán odečtením IPINu od zvoleného PINu. Všechny operace sčítání a odčítání se provádějí na jednotlivých číslicích (modulo 10) a k decimalizaci se používá decimalizační tabulka. Názorný příklad výpočtu zákazníkem zvoleného PINu při verifikaci metodou IBM 3624 Offset je uveden níže.

[16] Například IBM CCA API, Compaq-Atalla API a Thales-Zaxus-Racal API.

[17] Vstupem funkce nemůže být přímo PIN, protože bankovní programátor by mohl snadno vyzkoušet všechny PINy (10^4 možností) a určit hodnotu PINu v EPB.

Číslo účtu je reprezentováno pomocí ASCII číslic v dekadické soustavě a poté interpretováno jako hexadecimální vstup blokové šifry DES (resp. 3DES). Po zašifrování PIN generujícím klíčem je výsledek opět převeden do hexadecimální soustavy a zkrácen na první čtyři cifry. Ty však mohou obsahovat hodnoty $'\mathsf{A}'-'\mathsf{F}'$, které nejsou obsaženy na numerické klávesnici bankomatu, a proto jsou pomocí decimalizační tabulky převedeny na číslice dekadické soustavy. K výsledku je přičten offset, čímž se získá hodnota pro porovnání s PINem, který byl zadán bankomatu.

```
                    PAN:  4556 2385 7753 2239
          Zašifrovaný PAN:  3F7C 2201 00CA 8AB3
Zkrácený zašifrovaný PAN:  3F7C
   Hexadecimální číslice:  0123456789ABCDEF
   Decimalizacní tabulka:  0123456789012345
     Decimalizovaný IPIN:  3572
Verejne prístupný offset:  4344
  Zvolený ctyrmístný PIN:  7816
```

Tento postup byl použit v nejstarších typech bankomatů, a jsou dosti rozšířeny a implementovány i v nových HSM. Validační data byla tehdy společně s offsetem uložena na kartách s magnetickým proužkem, takže jediné, co musel bankomat chránit, byl PIN generující klíč. V současné době jsou tyto metody stále používány (podporuje je například i IBM CCA API), ale verifikace PINů již neprobíhá v bankomatu, ale ve vydávající bance.

Útoky využívající známých zašifrovaných PINů

Bez újmy na obecnosti předpokládejme, že jsou používány pouze čtyřmístné PINy. Nejjednodušším útokem je využití decimalizační tabulky ke zjištění číslic, které se vyskytují v PINu. Nastavíme-li například decimalizační tabulku na samé nuly, bude (bez použití offsetu) vždy po decimalizaci vygenerovaný PIN roven čtyřem nulám.

Tímto trikem můžeme pomocí funkce generující zašifrované PINy získat EPB obsahující PIN „0000“. Jestliže nyní při verifikaci použijeme jako parametry tento EPB a „nulovou“ decimalizační tabulku, proběhne verifikace úspěšně (tj. z EPB dešifrovaný PIN je roven vygenerovanému PINu). Nechť D_{orig} je korektní decimalizační tabulka a D_i jsou nové

binární decimalizační tabulky, které mají jedničku právě na těch pozicích, kde D_{orig} měla hodnotu i. Je-li například $D_{orig} =$ `0123456789012345`, pak $D_5 =$ `0000010000000001` a $D_9 =$ `0000000001000000`.

Nyní stačí, aby útočník pro i od 0 do 9 zavolal verifikační funkci s EPB nulového PINu a decimalizační tabulkou D_i. Není-li v hledaném PINu číslice i obsažena, změna v decimalizační tabulce se neprojeví a verifikace proběhne úspěšně. V opačném případě je i jedna z hledaných číslic PINu. K určení všech číslic vyskytujících se v PINu je potřeba provést verifikaci maximálně desetkrát. Celkově se tak počet možných kombinací PINů omezí z 10 000 na nejvýše[18] 36.

Druhá varianta útoku je efektivnější a navíc umožňuje přesně určit pořadí číslic v PINu. K její aplikaci je ale potřeba získat pět zašifrovaných hodnot známých PINů (`0000`, `0001`, `0010`, `0100`, `1000`). Protože bankovní systémy většinou neumožňují vkládání nezašifrovaných PINů a metodu z předchozího útoku nelze k vygenerování všech pěti hodnot PINů použít, je nutno získat je jinou cestou. Asi nejjednodušší je zadat tyto PINy bankomatu a zachytit je zašifrované poté, co dorazí do banky[19]. Jestliže se samotné hledání PINu implementuje, např. pomocí vhodně zkonstruovaného binárního stromu, který určí jakou decimalizační tabulku použít v dalším kroku, je možné nalézt PIN nejhůře na 24 pokusů, ale průměrně již na 15 pokusů [5].

Útok bez známého zašifrovaného PINu

Nutnou podmínkou předchozího útoku byla znalost EPB pro vybrané PINy. Následující útoky již toto nevyžadují. Předpokládejme, že se nám podařilo zachytit zákazníkův EPB obsahující správný PIN a že hodnota tohoto PINu ještě nikdy nebyla změněna (tj. offset je stále `0000`). Nechť D_{orig} je původní decimalizační tabulka a D_i jsou nové decimalizační tabulky. Platí, že D_i mají hodnotu $i - 1$ právě na těch pozicích, kde D_{orig} měla hodnotu i. Je-li například $D_{orig} =$ `0123456789012345`, pak $D_5 =$ `0123446789012344` a $D_9 =$ `0123456788012345`. Nyní stačí, aby útočník pro každou číslici i zavolal verifikační funkci se zachyceným EPB,

[18] V případě čtyřmístného PINu složeného z tří různých číslic. U čtyř různých číslic by kombinací bylo pouze 24 a u dvou číslic jen 14.

[19] To je zároveň nejjistější metoda, jak známý zašifrovaný PIN získat, protože použití funkce generující PINy bývá do značné míry omezeno (a její využití v prvním útoku bylo spíše ilustrativní).

správným offsetem (tj. `0000`) a decimalizační tabulkou D_i. Tímto způsobem, podobně jako u prvního útoku, zjistí číslice obsažené v zákazníkově PINu. Jejich pořadí pak dokáže určit vhodnou manipulací s offsety.

Uvažme běžný případ, kdy má zákazníkův PIN všechny číslice odlišné. Jako příklad zvolme PIN s hodnotou `1492` a pokusme se určit pozici číslice `2`. Hodnota PINu v EPB je vždy `1492`, ale hodnotu generovaného PINu lze použitím decimalizační tabulky D_2 změnit na `1491`. Tím se však docílí toho, že verifikace PINu neproběhne úspěšně. Nyní postupným voláním verifikační funkce s offsety `1000`, `0100`, `0010`, `0001` budeme naopak zvyšovat jednotlivé číslice generovaného PINu. Pouze v případě offsetu `0001` se však jeho hodnota vrátí zpět na `1492` a verifikace proběhne úspěšně. Použitím offsetu je pak jednoznačně určeno, která číslice PINu byla upravena. Ve skutečnosti se dokonce poslední verifikace ani nemusela provádět, protože nebyla-li hledaná číslice na předchozích třech pozicích, musela být na čtvrté.

Tímto způsobem může útočník určit pozice všech číslic, k čemuž v případě čtyřmístného PINu složeného z čtyř různých číslic potřebuje nejvýše 6 volání verifikační funkce (tři pro nalezení pozice první číslice, dvě pro nalezení pozice druhé číslice a jedno pro nalezení pozice třetí číslice). Poznamenejme, že tento útok je mírnou modifikací původního útoku z [5].

3.3.2 Check Value Attack

Jedná se o útok, který k získání PINů zneužívá funkci standardního API určenou k testování korektnosti zašifrovaných DES klíčů. Ta mimo jiné umožňuje šifrování 64bitového vzorku binárních nul. Zopakujme, že verifikační funkce zašifruje validační data PIN generujícím klíčem a výslednou hodnotu zkrátí a decimalizuje – tím je vytvořen IPIN. Testovací funkce tak umožňuje útočníkovi získat IPIN (stačí jen zkrátit a decimalizovat výsledek získaný testovací funkcí). Použitím verifikační funkce a manipulací s offsetem k danému EPB pak získá PIN.

3.3.3 ANSI X9.8 Attacks

Existence PIN-bloků dává možnost definovat příslušný formát. Výsledkem je, že existuje několik formátů, které jsou široce používány. Pro výrobce HSM to ovšem znamená nutnost implementovat několik funkcí

pro verifikaci PINů pro jednotlivé formáty a překladové funkce mezi formáty.

Z tohoto faktu vznikly útoky zneužívající špatného návrhu a implementace těchto funkcí používaných při verifikaci a překladu (změně formátů) PINů. Základním předpokladem těchto útoků je nízká entropie v PIN-blocích, jejímž důsledkem je nemožnost rozpoznat, který formát byl pro vytvoření daného PIN-bloku použit. To umožňuje zbavit se validačních dat v PIN bloku pomocí překladových funkcí (některé formáty je neobsahují), vložit validační data podle vlastního výběru, nebo se dostat k částem PIN-bloku, které by při použití jednoho formátu byly pro útočníka nedosažitelné. Detaily útoků jsou popsány v [14].

3.3.4 Další rozpracované útoky

Další dva útoky (PIN Derivation Attack a Collision Attack) byly objeveny Mikem Bondem, který se společně s Jolyonem Clulowem problematikou API na univerzitě v Cambridge zabývá. V době psaní tohoto příspěvku však ještě nebyly veřejně publikovány, byly s autory diskutovány prostřednictvím osobní e-mailové korespondence a informace o nich budou případně podány při prezentaci na konferenci EurOpen.

3.4 Nevynucení politiky – PKCS #11

Tato závěrečná část příspěvku se zabývá možnostmi, které útočníkovi nabízí nevhodné použití API, které neobsahuje a nevynucuje žádnou bezpečnostní politiku. Až doposud jsme se zabývali útoky, které byly aplikovatelné především na IBM CCA či jemu podobná kryptografická API. Ta byla navržena pro konkrétní kryptografické moduly a implementovala konkrétní bezpečnostní politiku. Přesto se jejich bezpečnost ukázala jako nedostatečná.

Nyní se zaměříme na velmi oblíbené rozhraní PKCS #11 [15], které je také často používáno jako hlavní API pro kryptografické moduly (implementace je i součástí operačních systémů). Oproti předchozím však bylo navrženo pouze jako standardní rozhraní mezi aplikacemi a jednouživatelskými bezpečnostními zařízeními. Hlavní problém tohoto API je, že je to pouze sada funkcí a neobsahuje žádnou politiku, která by například zajistila konzistentnost vlastností klíčů. Srovnání základních rysů CCA API a PKCS #11 pro IBM 4758 je uvedeno v [13]. Dále v této části budeme vycházet z [8, 15].

3.4.1 Základní informace o PKCS #11

Podle terminologie PKCS #11 jsou hardwarová bezpečnostní zařízení uchovávající objekty (např. data, klíče či certifikáty) a provádějící kryptografické operace nazývána *tokeny*. K jejich použití je nutné vždy vytvořit logické spojení s aplikací, což vyžaduje, aby se uživatel nejprve řádně přihlásil. Proběhne-li autentizace úspěšně, může pak prostřednictvím funkcí API s tokenem komunikovat. Uchovávané objekty se dělí podle jejich *viditelnosti* a *životnosti*. *Token objects* jsou stálé objekty, které jsou viditelné všem přihlášeným aplikacím s dostatečnými právy. *Session objects* jsou oproti tomu pouze dočasné objekty, které přetrvávají během vytvořeného spojení a jsou viditelné jen pro aplikaci, která je vytvořila.

Každý objekt je dále asociován s množinou vlastností, které popisují jeho typ a určují jeho použití. Například objekt klíč je vždy typu veřejný, soukromý či tajný, přičemž poslední dva z těchto typů mohou být navíc označeny jako *citlivé* či *neextrahovatelné*. Klíč, který je označen jako citlivý, nemůže být nikdy v otevřené podobě exportován mimo token. Neextrahovatelný klíč pak nemůže být exportován ani když je zašifrován. Tyto vlastnosti však nejsou s klíčem nijak kryptograficky svázány a importující aplikace si je tedy může libovolně upravit.

Oproti CCA API definuje PKCS #11 jen dva typy uživatelů: normální uživatele a bezpečnostní úředníky. Normální uživatel má po autentizaci možnost přistupovat k jednotlivým objektům a využívat kryptografických funkcí tokenu. Bezpečnostní úředník je zodpovědný za inicializaci tokenu a počáteční nastavení uživatelského hesla či PINu. Na rozdíl od normálních uživatelů nemůže provádět žádné kryptografické operace. Cílem PKCS #11 je poskytnout uchovávaným objektům dostatečnou ochranu před odhalením (např. označením klíčů jako citlivé a neextrahovatelné), ale není záměrem chránit objekty jednoho uživatele před použitím jinými uživateli.

3.4.2 Symmetric Key Attacks

V de facto nezměněné podobě lze na PKCS #11 aplikovat útoky *Key Conjuring* i *Meet in the Middle*. Další útoky pak většinou zneužívají podobných nedostatků jako útoky na CCA API.

3DES Key Binding Attack

Nedostatečná vazba mezi jednotlivými polovinami 3DES klíče umožňuje provést útok na každou jeho polovinu zvlášť. Označme požadovaný klíč jako K a jeho jednotlivé poloviny jako K_1, K_2. Při exportu klíče je každá polovina nezávisle zašifrována pomocí KEK (key encryption key) a platí tedy, že $E_{KEK}(K) = (E_{KEK}(K_1), E_{KEK}(K_2))$. Jednotlivé poloviny pak lze nezávisle na sobě importovat jako standardní DES klíč a zašifrovat jimi nějaký testovací vzorek. K jejich nalezení (hledá-li útočník oba klíče současně) pak stačí prohledat prostor přibližně 2^{56} klíčů.

Oproti předchozím API se v tomto případě útočník nemusí obávat problémů s kontrolou přístupu. Abychom předešli tomuto útoku, API by nemělo umožňovat, aby byl exportovaný klíč modifikován. Toho lze dosáhnout například použitím MAC[20].

Key Separation Attack

Jak již bylo zmíněno dříve, PKCS #11 specifikuje pro každý objekt typu klíč množinu vlastností, které určují k čemu smí být použit. Bezpečnostní chybou API je, že umožňuje konfliktní nastavení těchto vlastností a umožňuje tak kompromitování klíčů. Je-li například klíč označen jako KEK a zároveň jako klíč určený k dešifrování dat, lze jeho pomocí exportovat z tokenu libovolný extrahovatelný chráněný klíč a poté jej jednoduše jako data dešifrovat. Abychom předešli tomuto útoku, měla by být v API volba vlastností objektů mnohem více omezující. Tyto vlastnosti by navíc měly být s daným objektem nějak kryptograficky svázány.

Weaker Key/Algorithm Attack

PKCS #11 umožňuje zašifrování libovolného klíče pomocí algoritmů používajících klíč krátké délky (např. RC2 či DES). Útočník pak například nejprve exportuje požadovaný 3DES klíč K zašifrovaný pomocí standardního DES KEK (tj. $E_{KEK}(K)$). Poté exportuje KEK pomocí sebe sama (tj. $E_{KEK}(KEK)$) a útokem hrubou silou zjistí jeho hodnotu. Pomocí známého KEK pak již snadno získá hodnotu klíče K. Tímto je zcela degradována bezpečnost silných kryptografických algoritmů, a API by

[20] Message Authentication Code – tvoří podtřídu klíčovaných hašovacích funkcí a jejich cílem je zajištění integrity a autenticity zpráv.

proto vůbec nemělo k exportu klíčů použití slabších algoritmů umožňovat.

Related Key Attack

Podobně jako útok *Meet in the Middle* lze i tento útok aplikovat na klíče, které jsou označeny jako neextrahovatelné. Podívejme se nejprve, jak lze pomocí páru souvisejících klíčů $K_1 = (K_A, K_B, K_C)$ a $K_2 = (K_A \oplus \mathrm{DELTA}, K_B, K_C)$ učinit tří-klíčový 3DES jen nepatrně silnější než standardní DES. Útočník pouze zašifruje testovací vzorek P klíčem K_1 a dešifruje klíčem K_2, čímž získá:

$$\begin{aligned} P' &= D_{K_A \oplus DELTA}(E_{K_B}(D_{K_C}(E_{K_C}(D_{K_B}(E_{K_A}(P)))))) = \\ &= D_{K_A \oplus DELTA}(E_{K_A}(P)). \end{aligned}$$

Tím zcela izoloval část K_A, kterou teď může hrubou silou hledat nezávisle na částech K_B a K_C. Prohledávaný klíčový prostor se tím redukoval průměrně na 2^{55} a k nalezení K_A bude potřeba provést průměrně 2^{56} operací DES. Použije-li útočník 2^{16} souvisejících párů klíčů, pak lze aplikací *Meet in the Middle* útoku prohledávaný klíčový prostor redukovat až na 2^{39}.

Reduced Key Space Attack

Jednou z možností, jak z existujícího klíče vytvořit nový klíč, je výběr části jeho bitů (jedna z funkcí umožňuje vytvořit nový klíč výběrem části bitů z klíče existujícího). Ukažme například, jak lze použitím této metody z 32bitového tajného klíče s hodnotou `0x329F84A9` vytvořit nový 16bitový tajný klíč. `0x329F84A9` je binárně zapsáno jako `0011 0010 1001 1111 1000 0100 1010 1001` a jako počáteční pozici, od níž začne výběr nového klíče, stanovme bit b_{21}. Následujících 16 bitů $b_{21} \ldots b_{31} b_0 \ldots b_4$ pak vytvoří binární řetězec `1001 0101 0010 0110`, kterým je jednoznačně určena hodnota nově vytvořeného klíče jako `0x9526`.

Toho lze snadno využít ke zmenšení prohledávaného prostoru klíčů. Útočník např. nejprve použitím čtyřiceti po sobě jdoucích bitů z 56bitového DES klíče vytvoří 40bitový RC2 klíč (taková změna typu klíče je možná). Ten pak hrubou silou dešifruje a s jeho pomocí najde zbývajících 16 bitů původního DES klíče. Tento útok lze opět aplikovat i na neextrahovatelné klíče.

3.4.3 Public Key API Attacks

V následující části demonstrujeme útoky, které se opírají o podporu API pro kryptografické operace s veřejným klíčem a umožňují kompromitování soukromých nebo tajných klíčů.

Small Public Exponent with No Padding Attack

Problémem tohoto API je, že umožňuje používání funkcí, které už jsou zastaralé. Byl-li k šifrování klíče použit asymetrický algoritmus RSA bez doplnění (tzv. X.509 Raw RSA), je tento klíč z řetězce znaků pouze překonvertován na číslo, zašifrován a tento výsledek je opět převeden zpět na řetězec znaků. Je-li tedy veřejný RSA klíč dvojice m a e, lze operaci šifrování zapsat jako $C = K^e \bmod n$. Tato metoda je však v případě použití malé hodnoty veřejného exponentu napadnutelná, protože pokud $K^e < n$, tak je klíč možno jednoduše dešifrovat jako $K = C^{1/e}$. Je-li například požadováno, aby měl veřejný klíč z důvodů zvýšení rychlosti umocňování nízkou Hammingovu váhu, je vygenerování klíče s malou hodnotou exponentu poměrně pravděpodobné[21]. Tomuto útoku se dá snadno předejít zakázáním používání malých hodnot exponentu nebo použitím RSA s doplněním (tzv. PKCS #1 RSA).

Trojan Public Key Attack

Protože PKCS #11 ukládá veřejné klíče bez jakýchkoliv dalších integritních či autentizačních informací, může útočník snadno do tokenu vložit vlastní veřejný klíč. Jeho pomocí zašifruje a vyexportuje klíče, které pak pomocí svého soukromého klíče lehce dešifruje. Tímto jednoduchým útokem lze kompromitovat symetrické i asymetrické klíče, které nejsou označeny jako neextrahovatelné. API by tedy před použitím veřejného klíče k exportu citlivých informací mělo být schopno přinejmenším ověřit jeho původ.

Trojan Wrapped Key Attack

Podobně jako u předchozího útoku neumožňuje PKCS #11 zjistit ani původ libovolného zašifrovaného klíče. Obsahuje-li příslušný token řádný

[21] Například CCA API umožňuje generování asymetrických klíčů přímo s hodnotami veřejného klíče 3 nebo $2^{16} + 1$.

veřejný a soukromý klíč, může útočník jednoduše importovat svůj vlastní symetrický tajný klíč. Stačí jej nejprve zašifrovat veřejným klíčem a následně importovat. Tento klíč pak lze použít k exportu jiných klíčů ze zařízení a následně k jejich dešifrování. API by tedy mělo umožňovat ověřit i původ zašifrovaných klíčů určených k importu.

Private Key Modification Attack

V PKCS #11 mohou být soukromé klíče exportovány či importovány pouze tehdy, obsahují-li kromě soukromého exponentu a modulu také veřejný exponent a koeficienty CRT (tj. n, p, q, e, d, $d \bmod p-1$, $d \bmod q-1$ a $q-1 \bmod p$). Nyní uvažme situaci, kdy je soukromý RSA klíč zašifrován nějakým symetrickým tajným klíčem a exportován. Šifrování probíhá pomocí modu CBC a modifikace jednoho zašifrovaného bloku tedy způsobí změnu dvou bloků zašifrovaných dat. Protože celková délka zašifrovaných dat závisí především na velikosti asymetrických klíčů (typicky 512, 1 024 nebo 2 048 bitů), je pravděpodobné, že modifikace zašifrovaného bloku ovlivní nezávisle na ostatních datech pouze hodnotu samotného klíče. Jeho importováním pak získá útočník v tokenu částečně změněný klíč, který může použít k provedení útoku analýzou chyb [6]. V ideálním případě by tedy u soukromých klíčů měla být zajištěna integrita (např. použitím MAC nebo prováděním základních aritmetických testů typu $\mathrm{d} \equiv \mathrm{e}^{-1} \bmod n$ a $n = pq$).

4 Závěr

Příspěvek vznikl na základě (a podrobnější informace lze získat v) [14] – diplomové práce J. Krhovjáka, vedené V. Matyášem a oponované D. Cvrčkem.

Seznámili jsme se s bezpečnostní problematikou a základní architekturou HSM a byly představeny bezpečnostní požadavky na zařízení splňující stěžejní normu pro tuto oblast – FIPS 140-2 [11].

Hlavním cílem celého příspěvku byla prezentace útoků na a přes aplikační programovací rozhraní. Mnohé z těchto útoků byly aplikovatelné i na v dnešní době běžně používané IBM CCA API. Naší pozornosti neuniklo ani velmi oblíbené aplikační programovací rozhraní PKCS #11, ale jeho specifikace bohužel ponechává mnoho implementačních detailů na konkrétním výrobci HSM, což analýzu jednotlivých útoků dosti ztížilo.

I přesto, že cílem tohoto příspěvku nebylo stanovit, jak by měl bezpečný návrh kryptografických API vypadat, je porozumění chybám ve stávajících API prvním krokem, jak toho dosáhnout.

Literatura

[1] Anderson R. J.: *The Correctness of Crypto Transaction Sets.* In: *Proceedings of 8th International Workshop on Security Protocols*, volume 2133 of *Lecture Notes in Computer Science*, Springer, April 2000, pp. 125–127.

[2] Anderson R. J., Bond M.: *API-Level Attacks on Embedded Systems.* In: *IEEE Computer*, volume 34, October 2001, pp. 67–75.

[3] Bond M.: *A Chosen Key Difference Attack on Control Vectors*, November 2001.

[4] Bond M.: *Attacks on Cryptoprocessor Transaction Sets.* In: *Cryptographic Hardware and Embedded Systems*, volume 2162 of *Lecture Notes in Computer Science*, Springer, 2001, pp. 220–234.

[5] Bond M., Zieliński P.: *Decimalisation Table Attacks for PIN Cracking.* Technical Report 560, University of Cambridge, Computer Laboratory, February 2003.

[6] Boneh D., DeMillo R. A., Lipton R. J.: *On the Importance of Checking Cryptographic Protocols for Faults.* In: *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, Springer, December 1997, pp 37–51.

[7] Clulow J. S.: *PIN Recovery Attacks.* Technical Report 0520 00296, Prism, October 2001. Revised, October 2002.

[8] Clulow J. S.: *On the Security of PKCS #11.* In: *Cryptographic Hardware and Embedded Systems*, volume 2779 of *Lecture Notes in Computer Science*, Springer, 2003, pp 411–425.

[9] Clulow J. S.: *The Design and Analysis of Cryptographic Application Programming Interfaces for Security Devices.* Master's thesis, University of Natal, January 2003.

[10] *Federal Information Processing Standards Publication 140–1, Security Requirements for Cryptographic Modules.* National Institute of Standards and Technology, January 1994.

[11] *Federal Information Processing Standards Publication 140–2, Security Requirements for Cryptographic Modules.* National Institute of Standards and Technology, May 2001.

[12] *Federal Information Processing Standards Special Publication 800–29, A Comparison of the Security Requirements for Cryptographic Modules in FIPS 140–1 and FIPS 140–2.* National Institute of Standards and Technology, June 2001.

[13] IBM: *Comparison of CCA and PKCS #11 v2.01 for the IBM 4758.* White paper.

[14] Krhovják J.: *Analýza útoků na aplikační programovací rozhraní pro hardwarová bezpečnostní zařízení.* Master's thesis, Masaryk University Brno, 2003. Available at http://www.fi.muni.cz/∼xkrhovj/apinf/sdipr/DP_upravena_v1.pdf.

[15] RSA Security Inc. *Cryptographic Token Interface Standard – PKCS #11, Version 2.11.* November 2001.

CobiT® metodika pro řízení a správu procesů ICT

Luděk Novák, Kamil Golombek, Pavel Krečmer

E-mail: novak@isaca.cz, kamil.golombek@bdo-it.com, pavel.krecmer@bdo-it.com

Abstrakt

CobiT® (Control Objectives for Information and releated Technology) je uznávanou metodikou, která napomáhá ve zvládání managementu informačních a komunikačních technologií, které jsou používány pro naplnění hlavních cílů a záměrů organizace. Využívání ICT je v CobiT® řízeno souborem všeobecně akceptovaných nejlepších praktik tak, aby využití informací a nasazení informačních a komunikačních technologií přispívalo k dlouhodobému rozvoji organizace, prohlubovalo strategické cíle a snižovalo existující rizika použití ICT. Pro tyto potřeby CobiT® definuje 4 základní domény, které odráží základní řídící mechanismy ICT: (1) Plánování & Organizace, (2) Akvizice & Implementace, (3) Dodávka & Podpora a (4) Monitorování.

1 Základní charakteristika metodiky CobiT®

Pro dosažení úspěchu při nakládání s informacemi nelze dále považovat řízení organizace a management ICT za dvě oddělené a odlišné disciplíny. Efektivní řízení organizace soustřeďuje individuální a skupinové znalosti a zkušenosti, sleduje a hodnotí výkonnost a poskytuje záruky naplnění strategických cílů. Oblast ICT musí být dnes nedílnou součástí řízení organizace.

Management ICT vyžaduje uvědomělou strukturu, která mapuje vzájemné vztahy mezi ICT procesy, ICT zdroji a informacemi vůči strategickým cílům organizace. Management ICT je nedílnou součástí úspěšného

řízení organizace, zajišťující efektivní a měřitelný pokrok v souvisejících procesech organizace. Management ICT umožňuje organizaci plné využití výhod, které plynou z hodnoty informací, a tím dovoluje maximalizovat přínosy, účinně využívat nabízené příležitosti a dosahovat konkurenčních výhod.

Řízení a správa procesů ICT – ICT Governance je struktura vztahů a procesů přímého řízení a zpětných vazeb s cílem dosáhnout strategických cílů organizace při managementu ICT.

Cílem dobré a vžité praxe managementu ICT je zajistit, aby informace a informační technologie podporovaly obchodní cíle organizace, zdroje ICT byly odpovědně využívány a rizika ICT byla minimalizována. Tyto praktiky formulují základ pro směrování činností ICT, které je možné všeobecně rozdělit na:

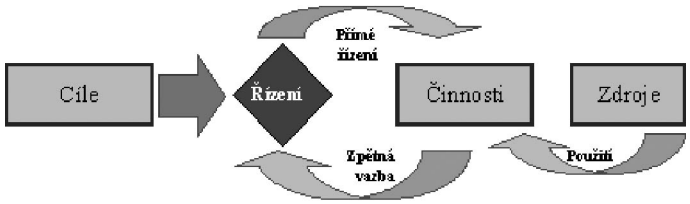
- Plánování & Organizace,
- Akvizice & Implementace,
- Dodávka & Podpora,
- Monitorování.

Primárním úkolem je minimalizace souvisejících rizik (prosazení informační bezpečnosti, zajištění spolehlivosti a shody) a maximalizace přínosů (růstem výkonnosti organizace a účelností vynaložených prostředků).

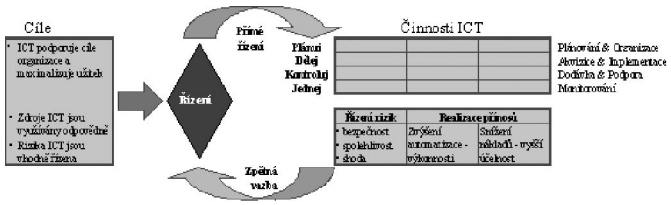
1.1 Základní model řízení

Obecný model řízení se opírá o prosazování cílů pomocí přímého řízení činností, kterých se účastní dostupné zdroje. U těchto činnostech musí být vytvořena intenzivní zpětná vazba, pomocí které je možné objektivně posoudit, zda provedení činností skutečně přispělo k naplnění daných cílů.

V některých systémech řízení je podobný princip označován jako model PDCA ze zkratek anglických slov Plan – Plánuj, Do – dělej, Check – kontroluj a Act – jednej. Na tomto modelu je založena nejen metodika CobiT® ale i řada dnes velmi známých systémů řízení např. systém řízení kvality podle ISO 9001 či systém řízení informační bezpečnosti podle BS 7799-2.



Obr. 1 Obecný model řízení organizace



Obr. 2 Základní model řízení a správy procesů ICT

Rozpracování těchto obecných modelů pro potřeby řízení a správy procesů ICT tak, jak je vnímá metodika CobiT® je zachyceno na obrázku výše. Z tohoto pohledu jsou jasně patrné čtyři základní domény, na které se metodika CobiT® soustředí.

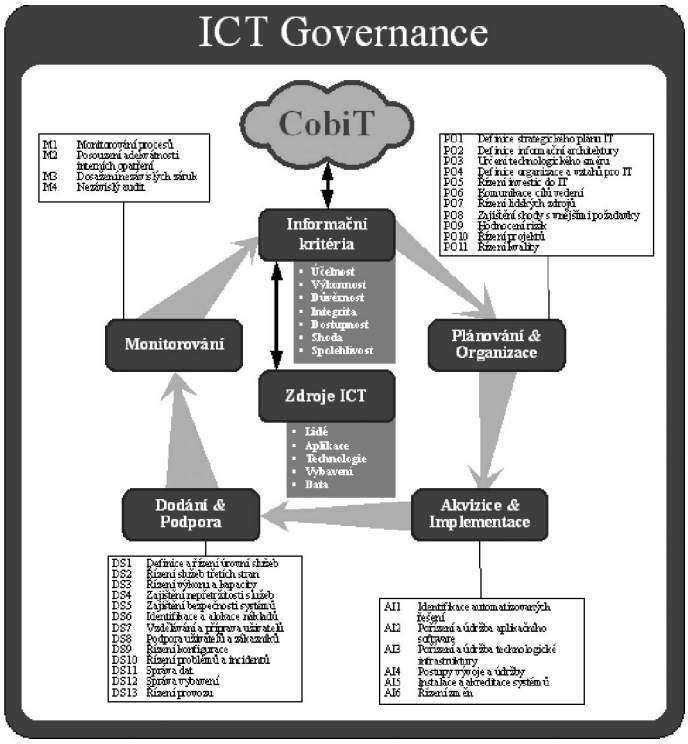
1.2 Základní domény ICT

CobiT® definuje 4 základní domény, které odráží základní řídící mechanismy ICT:

- **Plánování & Organizace (PO)** – tato doména se soustředí na definování strategie a hledání vhodných taktik pro dosažení nejvyšší míry přispění ICT k naplňování obecných cílů organizace.
- **Akvizice & Implementace (AI)** – aby bylo možné realizovat strategie ICT je důležité identifikovat, vyvinout nebo nakoupit a implementovat prostředky ICT, které jsou důsledně provázány s příslušnými procesy organizace.
- **Dodání & Podpora (DS)** – tato doména se soustředí na zajištění kvalitního dodání požadovaných služeb ICT, což zahrnuje činnosti od provozu přes bezpečnost až po přípravu uživatelů.

- **Monitorování (M)** – všechny procesy ICT musí být pravidelně hodnoceny, zda jejich kvalita provádění odpovídá stanoveným požadavkům a potřebám. Tato doména je zaměřena na získávání objektivního přehledu o prováděných činnostech ICT.

Širší pohled na tyto čtyři domény CobiT® je zachycen na následujícím obrázku. Z toho je patrné, že kromě těchto domén řízení jsou důležité ještě další dvě oblasti, kterými jsou informační kritéria a zdroje ICT.



Obr. 3 Procesy pro řízení a správu ICT

1.3 Informační kritéria

Základní požadavky na řízení a správu procesů ICT se dají charakterizovat pomocí následujících sedmi kategorií informačních kritérií. Jejich mírné překrývání se navzájem je možné:

Účelnost	se týká informací, které jsou důležité, a které se vztahují k obchodním procesům. Taktéž pokrývá doručení informací, které musí být provedeno včasným, správným, důsledným a obvyklým způsobem.
Výkonnost	se soustředí na zajištění, že informace jsou získány pomocí optimálního (nejvýhodnějšího a nejhospodárnějšího) použití zdrojů.
Důvěrnost	se soustředí na ochranu citlivosti informací před neautorizovaným prozrazením.
Integrita	se vztahuje k přesnosti a úplnosti informací a taktéž k jejich platnosti ve vztahu k obchodním hodnotám a nadějím.
Dostupnost	se týká toho, že informace, když je vyžadují obchodní procesy, jsou dostupné v dané chvíli i v budoucnosti. Též se soustředí na zabezpečení nezbytných zdrojů a přidružených schopností.
Shoda	se týká věrnosti těm zákonům, nařízením a smluvním vztahům, které jsou předmětem obchodních procesů tj. vnější obchodní kritéria.
Spolehlivost	se vztahuje k zajištění vhodných informací pro řízení práce s předměty a pro řízení uplatňování odpovědností za sledování financí a plnění.

1.4 Zdroje ICT

Zdroje ICT jsou v metodice CobiT® rozděleny do následujících pěti kategorií:

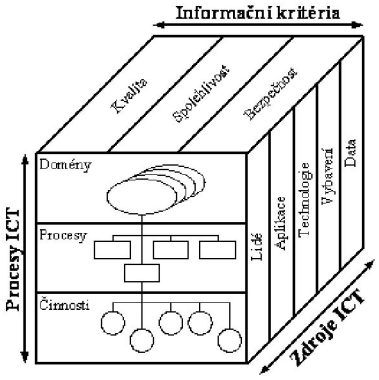
Lidé	zahrnuje schopnosti obsluhy, uvědomění, produktivitu, organizaci, nábor, podporu a sledování informačních systémů a služeb.
Aplikace	jsou vnímány jako soubor manuálních a naprogramovaných procedur.
Technologie	zahrnuje hardware, operační systémy, databáze, dohledové systémy, sítě, multimédia atd.
Vybavení	veškeré zdroje, které slouží k podpoře a umístění informačních technologií.
Data	objekty v nejširším slova smyslu (vnitřní i vnější), strukturované nebo nestrukturované, grafika, zvuk atd.

1.5 Krychle CobiT®

Na základě výše uvedených skutečností je možné tvrdit, že základní kostra metodiky CobiT® se soustředí na vzájemné vztahy mezi třemi klíčovými prvky. Za ty jsou považovány:

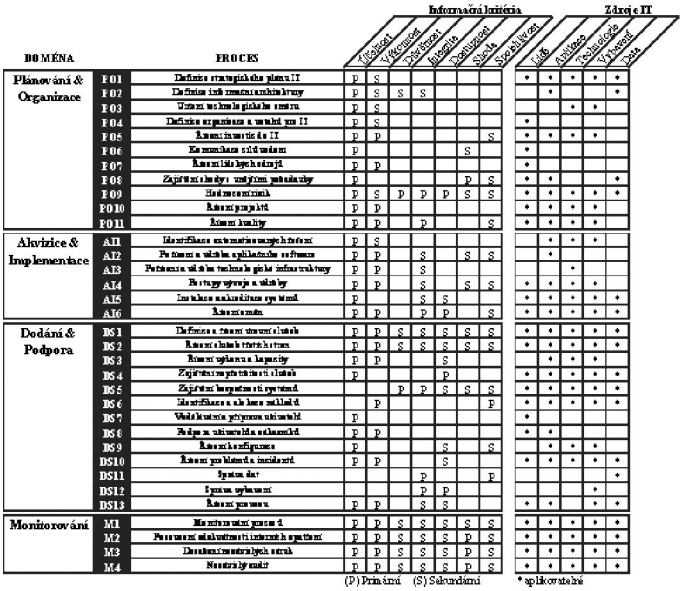
- **Informační kritéria** – vyjadřují požadavky, které jsou na ICT kladeny a podle kterých je posuzována úspěšnost naplnění stanovených cílů.
- **Zdroje ICT** – jsou dostupné prostředky, které je možné při řízení ICT využít.
- **Procesy ICT** – jsou skupiny činnosti, které jsou při řízení ICT vykovávány.

Vzájemné vztahy tvoří trojrozměrný prostor, který je znázorněn na následujícím obrázku.



Obr. 4 Krychle CobiT®

V následující tabulce jsou podrobně vyjádřeny všechny tři rozměry. Jednotlivé řádky obsahují 34 základních procesů řízení ICT. Ve sloupcích jsou indikována podstatná informační kritéria stejně jako aplikovatelné zdroje ICT.



DOMÉNA		PROCES	Informační kritéria: Účinnost	Efektivnost	Důvěrnost	Integrita	Dostupnost	Soulad	Spolehlivost	Zdroje IT: Lidé	Aplikace	Technologie	Vybavení	Data
Plánování & Organizace	PO1	Definice strategického plánu IT	P	S						•	•	•	•	•
	PO2	Definice informační architektury	P	S	S	S					•			•
	PO3	Určení technologického směru	P	S								•	•	
	PO4	Definice organizace a vztahů pro IT	P	S						•				
	PO5	Řízení investic do IT	P	P					S	•	•	•	•	
	PO6	Komunikace cílů vedením	P					S		•				
	PO7	Řízení lidských zdrojů	P	P						•				
	PO8	Zajištění shody s vnějšími požadavky	P					P	S	•	•			•
	PO9	Hodnocení rizik	P	S	P	P	P	S	S	•	•	•	•	•
	PO10	Řízení projektů	P	P						•	•	•	•	
	PO11	Řízení kvality	P	P		P			S	•	•	•	•	
Akvizice & Implementace	AI1	Identifikace automatizovaných řešení	P	S							•	•	•	
	AI2	Pořízení a údržba aplikačního software	P	P		S		S	S		•			
	AI3	Pořízení a údržba technologické infrastruktury	P	P		S						•		
	AI4	Postupy užívání a údržby	P	P		S		S	S	•	•	•	•	
	AI5	Instalace a akreditace systémů	P			S	S			•	•	•	•	•
	AI6	Řízení změn	P	P		P	P		S	•	•	•	•	•
Dodání & Podpora	DS1	Definice a řízení úrovní služeb	P	P	S	S	S	S	S	•	•	•	•	•
	DS2	Řízení služeb třetích stran	P	P	S	S	S	S	S	•	•	•	•	•
	DS3	Řízení výkonu a kapacity	P	P			S				•	•	•	
	DS4	Zajištění nepřetržitosti služeb	P				P			•	•	•	•	•
	DS5	Zajištění bezpečnosti systémů			P	P	S	S	S	•	•	•	•	•
	DS6	Identifikace a alokace nákladů		P					P	•	•	•	•	•
	DS7	Vhodné užívání výpočetních systémů	P							•				
	DS8	Podpora uživatelů a poradenství	P	P						•	•			
	DS9	Řízení konfigurace	P				S		S		•	•	•	
	DS10	Řízení problémů a incidentů	P	P			S			•	•	•	•	•
	DS11	Správa dat				P			P					•
	DS12	Správa vybavení				P	P						•	
	DS13	Řízení provozu	P	P		S	S			•	•		•	•
Monitorování	M1	Monitorování procesů	P	P	S	S	S	S	S	•	•	•	•	•
	M2	Posuzování shody s vnitřními systémy	P	P	S	S	S	P	S	•	•	•	•	•
	M3	Zhodnocení externích zdrojů	P	P	S	S	S	P	S	•	•	•	•	•
	M4	Nezávislý audit	P	P	S	S	S	P	S	•	•	•	•	•

(P) Primární (S) Sekundární * aplikovatelné

Tabulka 1 Přehled procesů řízení a správu ICT podle metodiky CobiT®

2 Metriky, indikátory a faktory

Management organizace potřebuje co nejpřesnější informace, s co nejvyšší objektivitou, na jejichž základě je možné efektivně řídit ICT a rizika s ním spojená. Aby bylo možné podat tyto objektivní informace, přichází CobiT® s metodou měření ICT, která stanovuje jak metodiku, tak měřítka nutná pro hodnocení.

2.1 Metrika modelu vyzrálosti

Pro vlastní orientaci v kvalitě ICT procesů je definován tzv. model vyzrálosti. Obecně popisují stav, ve kterém se organizace nachází, případně kam se chce v konečné podobě dostat. Model vyzrálosti se vyjadřuje číslem od 0 do 5 dle následující škály:

- **0 – Neexistuje** – nejsou aplikovány žádné řídící procesy. Organizace ani netuší, že by se něčím podobným měla zabývat.

- **1 – Inicializován/Náhodně** – procesy se řeší *ad-hoc* a nejsou nijak organizovány. Nicméně organizace si minimálně uvědomuje, že by tyto procesy měly být nějak řízeny. Hodně věcí se děje na základě individuální iniciativy.

- **2 – Opakovatelný, ale intuitivní** – procesy mají opakovatelný charakter. Tedy jestliže jsou prováděny různými lidmi, výsledky jsou podobné. Stále však neexistují žádné plány, odpovědnost zůstává na jedincích. Organizace je značně závislá na vědomostech a schopnostech těchto jedinců.

- **3 – Definovaný proces** – procesy jsou dokumentované a dobře sdělované, existuje systém vzdělávání a výcviku. Přesto dodržování těchto definovaných procesů závisí na rozhodnutí daného jedince a je vysoce pravděpodobné, že se nepodaří detekovat žádné odchylky od těchto procesů.

- **4 – Řízený a měřitelný** – procesy jsou sledovány a měřeny. Na základě měření a monitorování je možné odhalit odchylky a přijmout opatření tam, kde nejsou ICT procesy efektivní. Většina procesů však není automatizována.

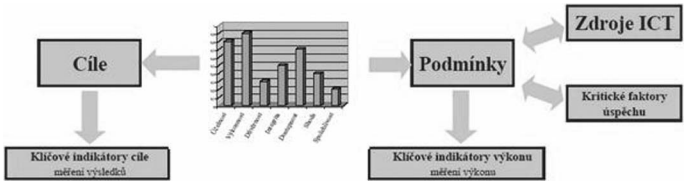
- **5 – Optimalizovaný** – jsou zavedeny a automatizovány nejlepší dostupné praktiky. Tato úroveň je odvozována v závislosti na porovnání ostatních organizací a nejlepších metod, které jsou používány. ICT jsou plně integrovány do života organizace, poskytuje kvalitní nástroje pro zajištění kvality a efektivnosti a způsobuje, že organizace je vysoce adaptabilní na změny okolí.

2.2 Indikátory a faktory

Dále CobiT® definuje několik základních faktorů, které slouží pro samotné řízení ICT. Jedná se o:

- **Kritické faktory úspěchu (CSF)** – definují nejdůležitější prvky nebo akce, které je třeba naplnit, aby daný základní cíl byl splnitelný. Definice jsou manažersky orientované a identifikují nejdůležitější prvky z pohledu strategického, technického, organizačního nebo procedurálního.

- **Klíčové indikátory cíle (KGI)** – jedná se o indikátory, které informují management, zdali ICT proces splnil očekávání organizace. Často se vyjadřují pomocí termínu jako dostupnost vyžadovaných informací, cenová efektivita, potvrzení shody, nízké nebo žádné rizika integrity nebo důvěrnosti atd. Podávají odpověď na otázku *čeho je třeba dosáhnout.*
- **Klíčové indikátory výkonu (KPI)** – určují, jak dobře podporuje daný ICT proces cíl organizace, jestli je pravděpodobně, že cíl bude/nebude dosažen, případně slouží jako dobré indikátory schopností, znalostí či zkušeností uživatelů. Na rozdíl od indikátorů klíčových cílů odpovídají na otázku *jak se daří dosahovat daný cíl.*



Obr. 5 Vztah mezi klíčovými indikátory cíle a výkonu

Vztah mezi indikátory je zachycen na 5. Základem vztahu je vazba mezi klíčovými indikátory cíle a klíčovými indikátory výkonu. Ty jsou navíc zásadně ovlivněny zdroji ICT a kritickými faktory úspěchu. Celý vztah lze shrnout v následující definici:

Kritické faktory úspěchu jsou ty nejdůležitější věci, které je nutno dělat podle volby vycházející z *modelu vyzrálosti*, a zároveň je nutné pomocí *klíčových indikátorů výkonu* sledovat, zdali byly naplněny základní úkoly, které jsou definovány *klíčovými indikátory cíle.*

3 Závěr

Metodika CobiT® je ucelenou a komplexní metodikou, která nachází použití u manažerů, uživatelů i auditorů ICT. Základní snahou metodiky je jasně strukturovat vysoce složitý systém řízení ICT a vybavit tento systém řízení vhodnými objektivními kritérii, podle kterých bude možné posuzovat úspěšnost či neúspěšnost jednotlivých oblastí řízení.

Ve smyslu úspěšného řízení ICT metodika CobiT® definuje následující čtyři obecné cíle řízení ICT:

- Přispět ke kvalitnímu fungování organizace,
- Orientovat se na zákazníka resp. uživatele,
- Zajistit provozní dokonalost,
- Orientovat se na možnosti rozvoje v budoucnosti.

Zajištění těchto obecných cílů není vůbec jednoduché a správné pochopení principů a vztahů, které jsou v metodice CobiT® zformulovány je velmi důležité pro zvládnutí tak nelehlého úkolu, jakým řízení ICT bezesporu je. Pomocí metodiky CobiT® na tento problém nezůstáváte samy, ale můžete využít zkušenosti mnoha odborníků z celého světa. Tak proč se alespoň nepoučit.

Literatura

[CobiT_ES] *CobiT 3rd Edition: Executive Summary, Information Systems Audit and Control Foundation.* ISACF 2000.

[CobiT_FW] *CobiT 3rd Edition: Framework, Information Systems Audit and Control Foundation.* ISACF 2000.

[CobiT_CO] *CobiT 3rd Edition: Control Objectives, Information Systems Audit and Control Foundation.* ISACF 2000.

[CobiT_MG] *CobiT 3rd Edition: Management Guidelines, Information Systems Audit and Control Foundation.* ISACF 2000.

[CobiT_AG] *CobiT 3rd Edition: Audit Guidelines, Information Systems Audit and Control Foundation.* ISACF 2000.

[CobiT_TS] *CobiT 3rd Edition: Implementation Tool Set, Information Systems Audit and Control Foundation.* ISACF 2000.

[CobiT_Sec] *CobiT Security Baseline: An Information Security Survival Kit, draft, IT Governance Institute.* ITGI 2004.

[CobiT_BB] *Board Briefing on IT Governance.* 2nd Edition, IT Governance Institute, ITGI 2003.

[CobiT_Map] *CobiT Mapping.* IT Governance Institute, ITGI 2004.

[CobiT_SO] *IT Control Objectives for Sarbanes-Oxley.* IT Governance Institute, ITGI 2003.

[ISACA] http://www.isaca.org.

[ITGI] http://www.itgi.org.

Luděk Novák (*1968) dokončil v roce 1991 Vojenskou Akademii v Brně, kde do roku 1994 působil jako odborný asistent. Poté pracoval na Generálním štábu Armády České republiky jako odborník na informační bezpečnost. Od poloviny roku 1999 se věnuje informační bezpečnosti a projektování bezpečnosti informačních systémů v komerčním sektoru. Nyní pracuje ve společnosti BDO IT, a. s. jako vedoucí konzultant. Autor je držitelem certifikátu CISA a členem odborných sdružení AFCEA a ISACA.

Kamil Golombek (*1974) ukončil v roce 1998 studium na Vojenské akademii v Brně. Svoji profesionální kariéru započal na Generálním štábu AČR, kde zastával různé INFOSEC funkce. Později přešel v rámci AČR do Vojenské zpravodajské služby jako INFOSEC specialista. Od roku 2002 pracuje jako konzultant v BDO IT, kde se zaměřuje především na oblast provádění bezpečnostních testů a auditů informačních systémů.

Pavel Krečmer (*1973) vystudoval katedru Speciálních komunikačních systémů Vojenské akademie v Brně, obor radiolektronický boj (1997). Do roku 2000 zastával různé pozice v AČR zaměřené na problematiku radioelektronického boje. Od roku 2000 pracoval jako INFOSEC specialista na Vojenském bezpečnostním úřadu Ministerstva obrany ČR se zaměřením na problematiku ochrany utajovaných skutečností v komunikačních a informačních systémech. V současnosti pracuje jako konzultant společnosti BDO IT. Zaměřuje se především na oblast bezpečnosti ICT, audit informačních systémů a bezpečnostní testy.

Zvýšení bezpečnosti softwarových aplikací pomocí kryptografické čipové karty

Petr Švenda,Vašek Matyáš

E-mail: xsvenda@fi.muni.cz, matyas@fi.muni.cz, t-matyas@microsoft.com

Abstrakt

Příspěvek se zabývá možností zvýšení bezpečnosti běhu softwarového agenta využitím programovatelné kryptografické čipové karty s podporou JavaCard a kombinací s metodami mobilní kryptografie. Popisuje ochranné rozhraní, umožňující přesun citlivých částí kódu do prostředí čipové karty, vzdáleně spravovatelné příkazy ve formátu XML. Je navržena speciální implementace autentizačního a transportního protokolu pro řízené využívání přesunutého kódu, která poskytuje ochranu autentizačním informacím uchovávaným na straně softwarového agenta a kódu provádějícího kroky protokolu i v případě, že je výpočetní prostředí pod kontrolou útočníka. Je využita implementace šifrovacího algoritmu AES, která umožňuje ukrýt hodnotu používaného klíče. Jsou navržena rozšíření zvyšující použitelnost této implementace v šifrovacím režimu CBC a umožňující větší provázanost s kódem softwarového agenta ve stylu mobilní kryptografie. Výsledný systém lze použít pro implementaci DRM architektur nebo kontroly přístupu k uživatelovým privátním informacím umístěným na čipové kartě, například při využití podpisového klíče.

1 Úvod

Digitální zpracování dat poskytuje mimo jiné možnost snadné tvorby identických kopií a rychlou distribuci bez ohledu na geografickou vzdálenost. Přináší s sebou však také problém kontroly jejich použití tak, aby byly zachovány zájmy všech zúčastněných stran. Efektivní správa práv k digitálním objektům (Digital Rights Managment, dále DRM) by měla tento cíl plnit. DRM, prováděná ve výpočetním prostředí koncového uživatele, bývá typicky zajištěna pomocí autonomní softwarové aplikace (dále softwarový agent) odpovědné za kontrolu využití digitálních dat v souladu s definovanými právy. Problémem běžných výpočetních prostředí typu PC je možnost útočníka plně kontrolovat běh tohoto softwarového agenta, především možnost čtení kódu a zpracovávaných dat a jejich následná modifikace. Útočník tak může docílit nežádoucí změny chování softwarového agenta úpravou jeho kódu anebo využít získaných dat k přímému přístupu k chráněným objektům.

Problémy související se zajištěním bezpečnosti běhu softwarové agenta lze částečně řešit použitím programovatelné kryptografické čipové karty, pokud poskytuje vyšší stupeň bezpečnosti než původní prostředí. Citlivý kód a data lze přenést a vykonávat na čipové kartě a využívat tak její hardwarové i softwarové ochranné mechanismy pro obranu před útočníkem. Vzhledem k omezeným výpočetním, paměťovým i funkčním (zobrazování dat apod.) prostředkům čipové karty však není možné ve většině případů přesunout celou funkčnost softwarového agenta, ale jen jeho vybrané části. Softwarový agent následně může využívat přesunuté části (dále chráněný algoritmus) pouze nepřímo, zasíláním vstupních dat a přijímáním zpracovaných výstupních dat. Pokud není požadováno omezení množiny softwarových agentů, které mohou chráněný algoritmus využívat a jde pouze o zajištění integrity výkonu nebo utajení implementace a důvěrnost použitých dat, je toto řešení postačující. Pokud je zapotřebí omezit množinu softwarových agentů oprávněných k využívání chráněného algoritmu, je třeba přistupující softwarové agenty vhodným způsobem autentizovat. Klíčovým problémem se stává uchování autentizačních informací na straně softwarového agenta, neboť ten se nachází v prostředí pod možnou kontrolou útočníka. Navíc musí být zajištěna i integrita celého procesu autentizace, neboť dochází i k autentizaci bezpečnostní proxy vůči softwarovému agentovi. V závislosti na povaze zpracovávaných dat může být navíc požadováno důvěrnost, integrita a čerstvost vyměňovaných zpráv.

V tomto příspěvku se věnujeme popisu systému adresujícího tento problém (dále ochranného rozhraní), pracujícímu v prostředí běžného PC s připojitelnou čipovou kartou podporující technologii JavaCard.

V dalším textu bude používáno následující označení zúčastněných stran: *Poskytovatelem* budeme označovat stranu, která implementuje funkčnost softwarového agenta, integruje ochranné rozhraní a vzdáleně ovládá celý systém. *Uživatelem* pak budeme označovat tu stranu, v jejímž výpočetním prostředí je softwarový agent vykonáván, a k němuž je připojena čipová karta. *Útočníkem* bude označen takový uživatel, který využívá (neomezené) kontroly nad svým výpočetním prostředím k ovlivnění chování softwarového agenta v rozporu se záměrem poskytovatele.

Dělení příspěvku je následující: Kapitola 2 nastiňuje problematiku ochrany softwarového agenta a uvádí vybrané typy ochran. Kapitola 3 popisuje základní strukturu ochranného rozhraní, poskytovanou funkčnost, možnosti vzdálené správy a modelový postup nasazení. V kapitole 4 jsou shrnuty základní vlastnosti implementace šifrovacího algoritmu AES navrženého v [CEJO02]. Kapitola 5 popisuje autentizační a transportní protokol navržený pro bezpečnou komunikaci mezi softwarovým agentem a čipovou kartou. V kapitole 6 jsou podrobněji rozebrány stavební prvky použité při implementaci tohoto protokolu. Kapitola 7 uvádí příklady scénářů, ve kterých je možné ochranné rozhraní využít.

2 Ochrana softwarového agenta

2.1 Typy útoků

Poskytovatel stojí před úkolem zabezpečit softwarového agenta tak, aby nedošlo k jeho zneužití útočníkem, který může své (prakticky) neomezené kontroly nad výpočetním prostředím softwarového agenta využít k následujícím typům útoků:

Zisk a modifikace kódu nebo dat – cílem je vytvoření mentálního modelu (části) softwarového agenta z jeho kódu pro následnou modifikaci jeho chování, zisk citlivých dat (šifrovací klíče) nebo extrakci obsažených návrhových myšlenek.

Tvorba diagramu toku dat – cílem útoku je rekonstrukce diagramu toku dat pro vybrané operace prováděné softwarovým agentem. Z vytvořeného diagramu a znalosti vstupních dat může útočník dedukovat informace o vnitřním stavu softwarového agenta a tento stav záměrně

modifikovat. Získaný diagram toku dat může být použit pro usnadnění tvorby mentálního modelu nebo modifikaci rozhodovacích podmínek.

Výkon kódu v nekorektním prostředí – cílem útoku je ovlivnění softwarového agenta jeho výkonem v prostředí s nekorektními vnějšími podmínkami. Příkladem nekorektních vnějších podmínek je nedostupnost požadovaných zdrojů (paměť), nekorektní formát vstupních dat nebo chybná funkčnost systémových funkcí včetně podvržení identity hostitelského systému. Může být dosaženo negativního omezení činnosti agenta potlačením funkcí kritických z hlediska poskytovatele (kontrola revokace, logování).

Manipulace komunikace – cílem útoku je narušení důvěrnosti, autenticity nebo čerstvosti komunikace softwarového agenta s druhou stranou. Útok je zaměřen na zisk přenášených dat (citlivé informace, šifrovací klíče) nebo na nekorektní chování softwarového agenta následkem manipulace vyměňovaných zpráv v průběhu komunikace.

2.2 Ochranné techniky

Pro ochranu softwarového agenta proti výše uvedeným útokům lze využít široké spektrum technik. Zde uvedené jsou děleny na čistě softwarové, které nevyužívají pomocná hardwarová zařízení a na ty, které výhod hardwarově chráněných zařízení využívají.

2.2.1 Softwarové ochrany

Z běžně rozšířených lze uvést využití registračního čísla měnícího se v závislosti na registračních údajích uživatele (jméno, firma) nebo hodnotách prostředí (sériová čísla harddisků, síťových karet, CPU), kontrola originálního média na očekávané vlastnosti (fyzické chyby, softwarové chyby, identifikační znaky). Pro detekci a eliminaci nástrojů využívaných útočníkem při provádění statické a dynamické analýzy (disassemblery, debuggery) nebo monitorování aktivit softwarového agenta (Regmon, Filemon) lze do jeho kódu umístit metody využívajících znalosti konkrétních nástrojů nebo obecných principů jejich chování. Dobrý přehled použitelných technik lze nalézt v [Ce02, Ze02].

Mezi pokročilejší techniky této skupiny patří nástroje pro automatickou transformaci kódu na sémanticky ekvivalentní, zároveň však ztěžující zpětnou tvorbu mentálního modelu (obfuskace). Výpočetní transformace vkládají do funkčního kódu mrtvý nebo irelevantní kód, rozši-

řují podmínky skoků o tautologie, odstraňují standardní programovací vzory, zavádějí redundantní operace, paralelizují sekvenční kód, agregují nesouvisející bloky kódu nebo nahrazují volání knihovních funkcí přímo jejich kódem. Datové transformace mění kódování dat, rozdělují a spojují proměnné, restrukturalizují datová pole a dynamicky generují statická data. Zároveň zavádějí transformace, které zvyšují paměťovou a výpočetní náročnost u nástrojů určených pro automatické odstranění provedené obfuskace. Principy základních obfuskačních metod lze nalézt v [CTL97, CTL98, CGJZ01, NCJ01].

Pro zamezení inspekce kódu a cílené modifikace lze použít šifrování kódu agenta klíčem ukrytým v jeho nešifrované části nebo odvozeným z předem známých charakteristik výpočetního prostředí. Dešifrování může být prováděno najednou během zavádění do operační paměti, nebo je dešifrována pouze aktuálně vykonávaná část (metoda plovoucího dešifrujícího „okna“). V prvním případě je vliv na rychlost běhu zanedbatelný, útočník však může využít nástrojů pro ukládání paměti a získat tak dešifrovanou podobu softwarového agenta. Ve druhém případě je v paměti vždy jen malá část celkového kódu, cenou je však snížení celkového výkonu následkem opakovaného dešifrování stejných částí kódu. Vhodnou volbou velikosti plovoucího okna lze nalézt kompromis mezi bezpečností a požadovaným výkonem softwarového agenta. Kritickým místem je v obou případech ukrytí metody pro generování a použití šifrovacího klíče. Slibnou metodu, umožňující výkon „šifrovaného“ kódu bez nutnosti jeho dešifrování, poskytuje pro některé třídy funkcí technika mobilní kryptografie prezentovaná v [SaTs98]. Je využívána popisovaným ochranným rozhraním a více popsána v kapitole 4. Při použití mobilní kryptografie lze přesněji stanovit výpočetní obtížnost odstranění ochrany.

Pokud je třeba chránit kód softwarového agenta pouze do doby jeho prvního použití, lze využít techniku neinformovaných agentů. Softwarový agent aplikuje dvakrát jednosměrnou hashovací funkci H na zvolenou událost v prostředí (jméno cílového počítače) a porovnává ji s nesenou hodnotou I. V případě shody použije hodnotu získanou pouze jednou aplikací funkce H k odvození šifrovacího klíče K. Při použití bezpečné funkce H nemůže útočník ze znalosti I odvodit hodnotu K. Další varianty použití lze nalézt v [RiSch98].

Na myšlence opakované kontroly integrity částí kódu jsou založeny ochrany proti modifikaci kódu prezentované v [ChaAt01, HMST01]. Kon-

trola je prováděna pomocí velkého počtu (řádově stovky) malých kusů kódu nazývaných testery, které co nejméně nápadně ověřují integritu přidělené části kódu. Při zjištění modifikace spouštějí reakční mechanismus, který na situaci příslušně zareaguje, například násilným ukončení běhu nebo chybnou funkčností. Při změně kódu pak musí útočník deaktivovat i tester nebo reakční mechanismus odpovědný za kontrolu modifikované části. Při násobném překrývání testovacích oblastí je nutno deaktivovat hned několik testerů nebo reakčních mechanismů. Kód testeru a reakčního mechanismu se zároveň nachází také v kontrolované oblasti a testery se tak navzájem chrání proti modifikaci.

2.2.2 Pomocná hardwarová zařízení

Ochrany tohoto typu spoléhají na využití pomocného hardwarového zařízení, u kterého náklady na zisk chráněných informací nebo vytvoření kopie překračují hodnotu informace, která má být chráněna před útočníkem. Nabízená funkčnost a rozsah nasazení hardwarových zařízení se pohybuje od zabezpečeného datového nosiče (paměťové karty), přes zařízení schopné provádět vybrané operace na vloženými daty (hardwarové klíče, čipové karty), až po komplexní výpočetní platformu v rozsahu současných PC (Trusted Computing Group).

Do této kategorie spadá i navrhované ochranné rozhraní využívající čipovou kartu s podporou JavaCard. Ve srovnání s jednoduchými hardwarovými klíči typu „dongles“ (Rainbow Sentinel, Alladin HASP) přináší větší výpočetní výkon, možnost bezpečné vzdálené aktualizace umístěných programových balíků prostřednictvím rozhraní OpenPlatform [OP02] a bezpečné sdílení výpočetního prostředí více stranami. Ochranné rozhraní neposkytuje mnohé z možností nabízených výpočetní platformou TCG [TCG04, An03], narozdíl od ní však nevyžaduje od uživatele nákladný přechod na specializovaný hardware. S výhodou lze využít běžně rozšířené kryptografické čipové karty distribuované za jiným účelem.

3 Ochranné rozhraní

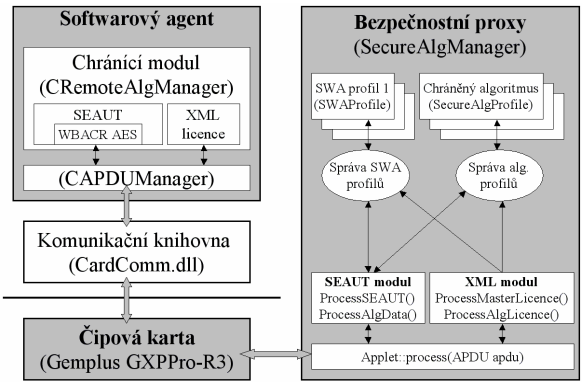
Ochranné rozhraní využívá spolupráce softwarového agenta s čipovou kartou s podporou JavaCard a skládá se ze dvou částí. Část umístěná na čipové kartě obsahuje jednoduchý XML parser, základní kostru (ro-

dičovský objekt) pro implementaci chráněných algoritmů a především funkčnost bezpečnostní proxy. Bezpečnostní proxy je jediný objekt, se kterým mohou softwaroví agenti přímo komunikovat prostřednictvím APDU příkazů. Veškerá komunikace mezi softwarovým agentem a chráněným algoritmem je zprostředkována přes tuto proxy, která na základě definovaných pravidel povoluje nebo zamítá jejich využití a zajišťuje zabezpečení komunikace. Část integrovaná do kódu softwarového agenta obsahuje funkčnost nutnou pro ustanovení komunikačního kanálu s bezpečnostní proxy, vzájemnou autentizaci a zasílání/zpracování dat vyměňovaných s chráněným algoritmem. Jako samostatná aplikace nebo součást softwarového agenta je přítomen modul pro předávání řídících příkazů ve formátu XML zasílaných poskytovatelem pro řízení bezpečnostní proxy. Implementace tohoto modulu nevyžaduje žádné zvláštní zabezpečení před útočníkem, neboť pouze předává komunikaci zabezpečenou na vyšší úrovni a nedisponuje žádnou tajnou informací.
Klíčové prvky ochranného rozhraní:

- Využití chráněného výpočetního prostředí programovatelné kryptografické čipové karty.
- Návrh způsobu autentizace a komunikace mezi čipovou kartou (bezpečnostní proxy) a softwarovým agentem zohledňující fakt, že softwarový agent je vykonáván v prostředí pod možnou kontrolou útočníka.
- Sdílení prostředí čipové karty více chráněnými algoritmy.
- Dynamická definice množiny softwarových agentů oprávněných k využívání konkrétního chráněného algoritmu.
- Vzdálená aktualizace interních hodnot chráněných algoritmů.
- Vzdálená správa celého systému pomocí zpráv ve formátu XML.

3.1 Komunikace od softwarového agenta

Přenos citlivé části z kódu softwarového agenta je realizován obdobným způsobem, jakým je postupováno při tvorbě a využití běžné knihovní funkce. Požadovaná funkčnost je implementována v prostředcích jazyka JavaCard [JC22API] přetížením metody, kterou volá bezpečnostní proxy



Obr. 1 Základní schéma ochranného rozhraní

po vyhodnocení oprávněnosti žádosti softwarového agenta o využití chráněného algoritmu. Původní funkčnost obsažená v těle softwarového agenta je nahrazena voláním metodu ochranného rozhraní, která zajistí ustanovení komunikačního kanálu, zaslání vstupních dat a zpracování výstupní odpovědi.

3.2 Vzdálená správa

Správa bezpečnostní proxy a interních hodnot jednotlivých chráněných algoritmů je prováděna pomocí dávkových příkazů ve formátu XML. Utajení a integrita dávek je zabezpečena pomocí šifrovacích klíčů sdíleného mezi poskytovatelem a bezpečnostní proxy. Zpracování dávky probíhá přímo na čipové kartě pomocí jednoduchého XML parseru vycházejícího z [Br02] (režim SAX) z důvodu zajištění integrity parsování a utajení zpracovávaných hodnot.

3.2.1 Bezpečnostní proxy

Řídícími příkazy je prováděna kompletní správa bezpečnostní proxy. Zajišťují obecnou funkčnost nutnou pro komunikaci mezi softwarovým agentem a bezpečnostní proxy, nezávislou na konkrétní implementaci chráněných algoritmů. Jejich zpracování by nemělo být nutné měnit. Řídící příkazy umožňují:

- Vytvoření/modifikaci/zrušení profilu softwarového agenta.
- Vytvoření/modifikaci/zrušení profilu chráněného algoritmu.
- Modifikace množiny softwarových agentů oprávněných využívat konkrétní chráněný algoritmus.
- Modifikace šifrovacích klíčů používaných pro zabezpečení dávkových zpráv.

Pomocí řídících příkazů lze dynamicky vytvářet vytvářet a rušit unikátně identifikované instance implementovaných tříd chráněných algoritmů. Každý softwarový agent musí mít v bezpečnostní proxy vytvořen svůj profil. Ten obsahuje jeho unikátní identifikaci, hodnoty šifrovacích klíčů sdílených s bezpečnostní proxy a seznam chráněných algoritmů, které mohou být tímto softwarovým agentem využívány. Bezpečnostní proxy pomocí unikátní identifikace jednotlivých instancí zajišťuje předání vstupních dat cílovým chráněným algoritmům. Profilů pro softwarové agenty i chráněné algoritmy může být vytvořeno více, až do předem definovaného počtu daného paměťovými možnosti použité čipové karty. Je umožněno souběžné využití chráněného algoritmu více softwarovými agenty.

3.2.2 Chráněné algoritmy

XML parser je přístupný i pro aktualizaci interních hodnot jednotlivých chráněných algoritmů. Implementace zpracování je plně prováděna poskytovatelem, ochranné rozhraní pouze zajišťuje bezpečný přenos a předání XML parseru. Možným využitím je vzdálená definice dodatečných podmínek, za kterých je umožněno použití chráněného algoritmu i pro oprávněné softwarové agenty. Typickým příkladem je použití čítače, který je při každém využití chráněného algoritmu snižován. Chráněný algoritmus je proveden pouze tehdy, pokud je hodnota čítače kladná. Čítač lze zvyšovat pouze prostřednictvím XML licence distribuované poskytovatelem, čehož lze využít pro základ DRM architektury.

3.3 Otázka výkonu

Přenos citlivých částí do prostředí čipové karty má vliv na celkový výkon softwarového agenta. Nemusí se nutně jednat o snížení výkonu, neboť čipová karta může poskytovat hardwarovou akceleraci některých operací

chráněného algoritmu. Při použití běžné kryptografické čipové karty však výkon pravděpodobně omezen bude a je tedy nutno vhodně zvolit, která část kódu softwarového agenta bude chráněna. Na změnu výkonu mají vliv následující faktory: a) rychlost vytvoření autentizovaného kanálu dle SEAUT, b) propustnost datové vrstvy mezi softwarovým agentem a čipovou kartou, c) propustnost komunikační vrstvy dle SEAUT, d) rychlost přípravy dat pro chráněný algoritmus na straně softwarového agenta a e) rychlost provedení chráněného algoritmu nad vstupními daty. Pro konkrétní typ čipové karty se liší vliv jednotlivých faktorů. V současné době je významným faktorem především propustnost datové vrstvy díky omezené rychlosti komunikace prostřednictvím APDU příkazů. V závislosti na používaných operacích může být významným faktorem e). Pro zanedbatelný vliv faktorů a) a c) je na straně hardwarového tokenu vyžadována hardwarová podpora šifrovacího algoritmu AES.

Tabulka 1 zachycuje výsledky orientačního testu, provedeného za účelem odhadu doby trvání základní operací ochranného rozhraní. Pro odhad byla použita záměna šifrovacího algoritmu AES za algoritmus DES, hardwarově podporovaného použitou čipovou kartou Gemplus GXPPro-R3. Z porovnání rychlosti hardwarových implementací algoritmů DES a AES uvedených v [GaCho01, SaMo03] lze při hardwarové podpoře AES očekávat mírně lepší výsledky oproti uvedeným. Celkové zrychlení bude závislé na poměru šifrování a ostatních operací.

3.4 Dodatečné ochranné techniky

Konstrukce části ochranného rozhraní integrovaného v kódu softwarového agenta (především implementace komunikačního protokolu SEAUT) je navržena s cílem ztížit útočníkovi provedení smysluplné modifikace kódu, získání vyměňovaných dat a podvržení dříve zachycených zpráv vyměňovaných s bezpečnostní proxy. Cílem použitých technik je donutit útočníka vytvářet mentální model z co největší části kódu, v případě pokusu o modifikaci pak zvýšit počet míst nutných pro změnu chování. Representace kryptografických klíčů u softwarového agenta pomocí WBACR AES tabulek (viz kapitola 4) chrání jejich otevřenou hodnotu, nezabraňuje však útočníkovi v jejich použití pokud dojde k jejich extrakci. Proto je vhodné doplnit ochranné rozhraní o další ochranné prostředky. Vhodnými kandidáty jsou obfuskační nástroje pro ztížení extrakce nebo nahrazení WBACR AES tabulek, použití sebekontrolujícího kódu pro zajištění integrity a ochrany zaměřené proti standardním

Operace	**Čas**
Zaslání 1 APDU (0 B vstup, 0B výstup)	0,07 s
Zaslání 1 APDU (256 B vstup, 0B výstup)	0,16 s
Zaslání 1 APDU (256 B vstup, 256B výstup)	0,3 s
Použití algoritmu (0 B dat = 1 APDU)	0,36 s
Použití algoritmu (240 B dat = 1 APDU)	0,61 s
Použití chráněného algoritmu (1 kB dat = 4 APDU)	2,4 s
Navázání autentizovaného spojení dle SEAUT	0,52 s
Zpracování příkazu [NewAlg] (220 B = 1 APDU)	13,7 s
Zpracování příkazu [RemoveAlg] (130 B = 1 APDU)	6 s

Tabulka 1 Trvání vybraných operací ochranného rozhraní na čipové kartě Gemplus GXPPro-R3. Uvedené hodnoty jsou pouze orientační, neboť použitá karta nemá implementovánu hardwarovou podporu algoritmu AES. Hodnoty vycházejí z měření při použití algoritmu DES. Pro potřeby testu chráněný algoritmus nad vstupními daty neprovádí žádnou operaci, pouze je předá v nezměněné podobě na výstup. Příkaz [NewAlg] vytvoří nový profil chráněného algoritmu. Příkaz [RemoveAlg] odstraní profil chráněného algoritmu

statickým a dynamickým inspekčním nástrojům. Použitelnost konkrétních nástrojů nebyla zkoumána.

3.5 Postup nasazení

Následujících pět kroků shrnuje logické operace, které je potřeba učinit pro integraci a nasazení ochranného rozhraní.

1. Fyzická distribuce čipové karty – distribuovanou čipovou kartu lze pro potřeby ochranného rozhraní využít opakovaně. S výhodou lze použít čipových karet, které již uživatel vlastní, například SIM karty nebo podpisové čipové karty. Podmínkou je samozřejmě splnění funkčních a bezpečnostních požadavků.

2. Implementace chráněných algoritmů – druhým krokem je implementace chráněných algoritmů, překlad a (vzdálená) instalace bezpečnostní proxy na čipovou kartu. Instalovaný aplet, který přestane posta-

čovat požadavkům, může být vzdáleně odstraněn a opakováním druhého kroku nahrazen novým. Vzdálenou instalaci lze provést bezpečným způsobem prostřednictvím vhodného rozhraní podporovaného kartou, například specifikace OpenPlatform [OP02] podporující instalaci podepsaných a šifrovaných apletů. Bezpečnostní proxy je během instalace přiřazena unikátní identifikace.

3. Implementace softwarového agenta – dochází k začlenění volání metod ochranného modulu, přiřazení unikátní identifikace softwarového agenta a generování WBACR AES tabulek pro použité šifrovací klíče. Softwarový agent je přeložen do spustitelné podoby a distribuován k uživateli. Třetí krok může být dle potřeby opakován, pomocí následného čtvrtého kroku lze povolovat a omezovat využití služeb bezpečnostní proxy pro distribuované softwarové agenty.

4. Správa XML příkazy – čtvrtým krokem je tvorba řídících příkazů, které vytvoří profil softwarového agenta u cílové bezpečnostní proxy a povolí využití zvolených chráněných algoritmů. Lze vytvářet a distribuovat příkazy pro aktualizaci interních hodnot chráněných algoritmů. Čtvrtý krok je dle potřeby opakován. Příkazy jsou distribuovány v šifrované podobě zabezpečené klíčem sdíleným mezi bezpečnostní proxy a poskytovatelem.

5. Využití chráněného algoritmu – pátým krokem je využití chráněného algoritmu softwarovým agentem. Softwarový agent vytváří bezpečný komunikační kanál a zasílá bezpečnostní proxy požadavky. Bezpečnostní proxy rozhoduje o oprávněnosti požadavku na základě autentizace softwarového agenta a seznamu chráněných algoritmů, které může softwarový agent využívat. V kladném případě zasílá zpracovaný požadavek zpět softwarovému agentovi.

4 WhiteBox Attack Resistant AES

Pro ochranu hodnot šifrovacích klíčů a dat zpracovávaných softwarovým agentem je využito konceptu mobilní kryptografie, navrženém v [SaTs98]. Výpočtem s šifrovanou funkcí (Computing with Encrypted Function) je zde označován proces, při kterém strana B vykoná nad vlastními vstupními daty X program P, poskytnutý stranou A, realizující operaci F. Program P realizuje operaci F takovým způsobem, že strana B se během výkonu P nad X nedozví žádnou podstatnou informaci, která by sloužila k odhalení operace F.

Tato myšlenka je využita pro implementace šifrovacího algoritmu AES navrženém v [CEJO02] (dále WBACR AES). Umožňuje šifrovat v prostředí pod kontrolou útočníka bez vyzrazení hodnoty použitého šifrovacího klíče. Narozdíl od běžných implementací, přijímajících na vstupu data a hodnotu šifrovacího klíče, je přijímán pouze blok vstupních dat. Ten je modifikován pouze pomocí série náhledů do předpočtených tabulek. Po posledním náhledu je vstupní blok zašifrován klíčem, který byl použit pro generování tabulek. Šifrování využívající WBACR AES má oproti standardní implementaci několik předností:

1. Utajení hodnoty použitého klíče – zpětný zisk hodnoty klíče je výpočetně obtížný i při znalosti vygenerovaných tabulek. Průběžné výsledky během série náhledů jsou chráněny pomocí náhodné bijektivní transformace generované během tvorby tabulek.

2. Oddělitelná šifrovací a dešifrovací funkčnost – vygenerované tabulky pro šifrování a dešifrování jsou navzájem nezávislé. Softwarový agent tak může pro daný klíč obsahovat tabulky použitelné pouze pro šifrování nebo dešifrování. Díky této vlastnosti lze vyměňovaná data šifrovat pomocí symetrické kryptografie a zároveň využít výhod plynoucích ze dvou oddělených klíčů asymetrické kryptografie. Dešifrovací část tabulek může sloužit jako „veřejný“ klíč distribuovaný všem softwarovým agentům. Šifrovací část vlastní pouze jeden softwarový agent a slouží mu jako „privátní“ klíč. Proces vytváření digitálního podpisu zprávy může být nahrazen pomocí časově méně náročné tvorby autentizačního kódu (MAC).

3. Vstupní a výstupní kódování – vstupním kódováním je myšlena bijektivní transformace vstupního bloku dat, která je v rámci náhledů do předpočtených tabulek pro první rundu pomocí inverzní transformace (zahrnuté v tabulkách) odstraněna. Tato transformace je volena náhodně během generování tabulek. V případě identické transformace jsou vstupní data přijímána přímo, v opačném případě je třeba na ně před začátkem šifrování aplikovat tuto transformaci. Analogicky pro výstupní kódování aplikované v rámci náhledů poslední rundy, které je nutno před použitím výstupních dat inverzní transformací odstranit. Použití vstupního anebo výstupního kódování ztěžuje možnost samostatného použití tabulek extrahovaných z kódu softwarového agenta, neboť útočník musí získat i předpis použitých bijektivních transformací a zvyšuje celkovou provázanost kódu. Navíc poskytuje základ pro „napojení“ dalších operací ve stylu mobilní kryptografie.

5 SEcure AUthenticated Transport protokol

Pro zajištění vzájemné autentizace, důvěrnosti a čerstvosti vyměňované komunikace mezi bezpečnostní proxy a softwarovým agentem byl navržen protokol (Secure Authenticated Transport protokol, dále SEAUT), který zohledňuje umístění softwarového agenta v prostředí pod možnou kontrolou útočníka (prostředí bezpečnostní proxy je považováno za bezpečné). Běžné autentizační protokoly na bázi symetrické nebo asymetrické kryptografie nelze použít, neboť útočník má možnost na straně softwarového agenta pomocí ladících nástrojů číst hodnoty autentizačních informací, modifikovat hodnoty proměnných, narušovat integritu kroků použitého protokolu nebo využít autentizačních funkcí ve vlastní režii. Obdobné problémy vyvstávají při potřebě důvěrnosti a čerstvosti vyměňované komunikace.

Vhodná implementace použitého protokolu by měla splňovat následující podmínky: a) strana B neprovádí žádné porovnávací operace vzhledem k očekávaným hodnotám, b) autentizační informace strany B a informace sloužící k utajení vyměňovaných dat nejsou čitelné při použití statické a dynamické inspekce, c) strana B obsahuje robustní mechanismus kontroly čerstvosti, d) strana B je chráněna proti vytvoření mentálního modelu a modifikaci a e) strana B neobsahuje funkci generující zprávy zaměnitelné za zprávy pocházející od strany A, a to ani v případě, že útočník manipuluje se vstupními daty této funkce.

Navržený protokol SEAUT se skládá ze dvou částí, část autentizační a část transportní. Stranu umístěnou v bezpečném prostředí (bezpečnostní proxy) označme A, stranu umístěnou v prostředí pod možnou kontrolou útočníka (softwarový agent) označme B.

5.1 Autentizační část

Autentizační část navrženého protokolu je založena na 3-průchodovém protokolu ISO9798-2 (dále P3-ISO9798-2) pro vzájemnou autentizaci [ISO9798-2], využívajícím sdílený klíč K_S pro symetrickou kryptografii a náhodné číslo jako keksík pro zajištění čerstvosti.

Vyměňované zprávy (autentizační část):

1. $A \leftarrow B : \{id_B, N_B\}$.
2. $A \rightarrow B : \{id_A, E_{KD_A}(N_A, N_B, id_B)\}$.
3. $A \leftarrow B : \{E_{KE_A}(N_B, N_A)\}$.
4. $K_{R_{init}} = H_1(N_A|N_B|id_B|V(N_B)|V(id_B))$

Protokol SEAUT používá na místo jednoho klíče K_S klíče KE_A a KD_A. Pro šifrování 2. zprávy je použit klíč KD_A, pro 3. zprávu KE_A. Lze snadno nahlédnout, že strana B využívá klíč KE_A pouze pro zašifrování a klíč KD_A pouze pro dešifrování (opačně pro stranu A). Cílem této úpravy je zajistit, aby na straně B nemusela být přítomna funkčnost pro dešifrování klíčem KE_A. Klíč KE_A tak může být na straně B realizován pouze šifrovací částí WBACR AES tabulek (viz 4). Analogicky pro klíč KD_A.

Další modifikací je způsob vyhodnocení korektnosti provedené autentizace na straně B. Kontrola shody keksíku N_B a identifikace id_B vůči očekávaným hodnotám $V(N_B)$ a $V(id_B)$ z 1. a 2. zprávy je prováděna jen nepovinně, neboť ji útočník může snadno modifikovat nebo odstranit. Namísto toho je ze zaslaných i obdržených hodnot pomocí jednosměrné klíčované hashovací funkce H_1 (viz 6.2) vytvořena iniciální hodnota klíče relace K_R. Klíč relace K_R bude ustanoven i v případě nekorektní autentizace strany A vůči straně B, bude však odlišný od klíče vzešlého z korektní autentizace. Použití klíčované hashovací funkce (s využitím WBACR AES) pro jeho tvorbu zabraňuje útočníkovi odvodit jeho přímou hodnotu. Vzhledem ke způsobu použití K_R v transportní části pak nekorektní K_R povede chybnému zpracování vyměňovaných zpráv v případě, že útočník použije zprávy zachycené během předchozí (korektní) komunikace mezi A a B.

5.2 Transportní část

Transportní část protokolu zajišťuje důvěrnost a čerstvost vyměňovaných zpráv. K zachování důvěrnosti je využita dvojice šifrovacích klíčů KE_T a KD_T, realizovaných a používaných obdobně jako klíče KE_A a KD_A v autentizační části. Pokud strana B pomocí klíče KD_T korektně dešifruje příchozí zprávu, může si být jista, že pochází od strany A, neboť sama nedisponuje funkčností pro zašifrování tímto klíčem a útočník ji tak nemůže zneužít pro vytvoření podvržené zprávy. Ze stejného důvodu je chráněn před útočníkem obsah zachycených zpráv určených pro stranu A, neboť B nedisponuje funkčností pro jejich dešifrování.
Vyměňované zprávy (transportní část):

5. A, B vypočtou: $K_{R_i} = H_2(K_{R_{i-1}})$.
6. $A \leftarrow B : \{idA, idB, X_{K_{R_i}}(E_{KE_T}(request))\}$.
7. A, B vypočtou: $K_{R_{i+1}} = H_2(K_{R_i})$.
8. $A \rightarrow B : \{idA, idB, X_{K_{R_{i+1}}}(E_{KD_T}(response))\}$.

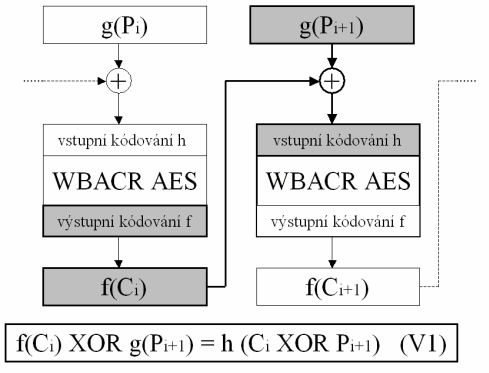
Problémem tak zůstává robustní zajištění čerstvosti (na straně B). Metody založené na porovnávacích kontrolách různých typů keksíku (náhodné číslo, čas, pořadové číslo) nelze použít z důvodu snadné manipulaci kontrolního kódu. Vzhledem k implementaci šifrovacího algoritmu pomocí WBACR AES nelze využít ani pravidelnou změnu šifrovacího klíče. Navržená implementace využívá klíče relace K_R, aktualizovaného pomocí klíčované hashovací funkce H_2 po každé odeslané i přijaté zprávě. Klíč K_R je použit takovým způsobem (operace X_{K_R}), aby ovlivnil každý bit odesílané i přijímané zprávy (viz 6.2). Jeho nekorektní hodnota vzhledem ke zpracovávané zprávě tak vede k jejímu poškození. Při použití vstupního a výstupního kódování pro aktualizaci K_R není na straně B jeho otevřená hodnota použita a je tak ztížena jeho lokalizace nebo modifikace.

WBACR AES tabulky pro šifrovací klíče KE_T a KD_T jsou generovány s využitím vstupního a výstupního kódování (narozdíl od KE_A a KD_A). Data získaná dešifrováním zprávy pomocí klíče KD_T tak nejsou ihned přístupná v otevřené podobě čitelné útočníkem. Odstranění použitého kódování může být provedeno postupně v následujícím kódu nebo může sloužit k napojení další operace ve stylu mobilní kryptografie. Zvyšuje se tím provázanost jednotlivých částí kódu, útočník má ztíženo získání dat v otevřené podobě a extrakci nebo nahrazení WBACR AES tabulek. Volbou různého vstupního a výstupního kódování lze provádět personalizaci softwarového agenta strany B nezávisle na straně A.

6 Stavební prvky SEAUT

6.1 I/O kódování pro CBC režim

Vstupní a výstupní kódování (dále IOC), jak je popsáno v [CEJO02], poskytuje způsob, jak ztížit použití WBACR AES tabulek, pokud dojde k jejich extrakci. Bez dodatečných úprav však IOC nelze použít pro jiný šifrovací režim než ECB. Použití pro režim CBC by vyžadovalo odstranění resp. aplikaci IOC před každou jeho iterací, což by výrazně snížilo rychlost šifrování. Zároveň by byl kód aplikující kódování s využitím dynamické inspekce snadno lokalizovatelný a jeho použití by ztratilo význam. Pro praktickou možnost využití IOC pro režim CBC byla navržena modifikace znázorněná na obrázku 2. Ke dvojici vstupního h a výstupního kódování f je přidáno datové kódování g. Narozdíl od původního však nejsou jednotlivá kódování nezávislá, ale jsou generována tak, aby



Obr. 2 Vstupní a výstupní kódování použitelné pro šifrovací režim CBC. P_i značí i-tý blok vstupních dat, C_i značí i-tý blok zašifrovaných dat

byl splněn vztah $f(C_i)\ XOR\ g(P_{i+1}) = h(C_i\ XOR\ P_{i+1})$. Po aplikaci funkce XOR na výstup předchozí iterace (kódování f) a následujícího vstupu (kódování g) získáme hodnotu $h(C_i\ XOR\ P_{i+1})$. Použití IOC během celého procesu šifrování (resp. dešifrování) je transparentní a nevyžaduje žádnou změnu oproti běžnému režimu CBC. Datové kódování g je aplikováno v libovolném místě před počátkem šifrování a může být součástí předchozí operace navržené ve stylu mobilní kryptografie. Analogicky pro výstupní kódování f.

Interní struktura WBACR AES prakticky umožňuje použít IOC o maximální velikosti 8 bitů. Nelze tedy použít jediné kódování pro celý 128 bitový šifrovaný blok. Operace XOR je ale „blokovatelná“ (výsledek i-tého bitu nezávisí na j-tém bitu), lze tedy použít 16 nezávislých IOC_1 až IOC_{16} pro 1 až 16 bajt bloku.

6.2 Použití a aktualizace K_R

Klíč relace K_R je vytvářen klíčovanou hashovací funkcí z očekávaných a obdržených hodnot během autentizační části protokolu SEAUT. Tvorba hashovací funkce z blokového šifrovače realizovaného s využitím WBACR AES zvyšuje bezpečnost K_R, neboť útočník nemůže bez extrakce klíče určit ve vlastní režii výslednou hodnotu K_R. Hashovací funkce má výstupní kódování odpovídající vstupnímu kódování K_R.

Hodnota K_R je použita dvěma způsoby:

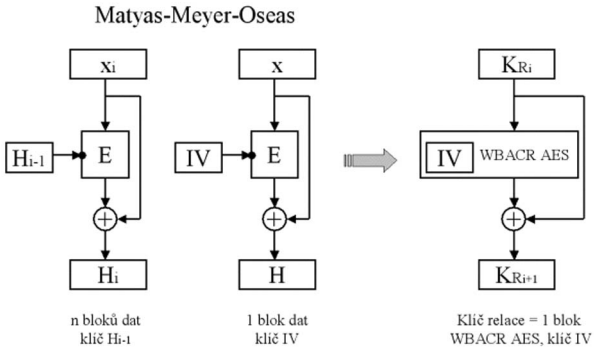
1. Inicializační vektor pro režim CBC – cílem je zajistit, aby chybně ustanovený klíč relace K_R vedl k vytváření a zpracování nekorektních zpráv vyměňovaných během transportní části. Pokud útočník modifikuje běh softwarového agenta tak, aby pokračoval i přes nekorektní autentizaci, bude hodnota K_R odlišná a vyměňované zprávy nebudou zpracovány korektně.

2. Opakovaná aplikace pomocí operace XOR na zašifrovaná data – cílem je zajistit, aby zpracování příchozí zprávy vytvořené pomocí jiné hodnoty K_R vedlo k chybnému zpracování celé zprávy, ne pouze prvního bloku jako v případě použití pro inicializační vektor.

Hodnota K_R je aktualizována po každé přijaté nebo odeslané zprávě. Navržená metoda aktualizace je založena na využití schématu Matyas-Meyer-Oseas (MMO) pro tvorbu hashovací funkce z blokového šifrovače, jak je znázorněno na obrázku 3. Vzhledem k fixní velikosti K_R dochází vždy pouze k jedinému cyklu MMO a hodnota použitého šifrovacího klíče se nemění. Tato vlastnost umožňuje realizovat blokový šifrovač šifrovací část9 WBACR AES tabulek. Navržená metoda zajišťuje vlastnost kryptografické jednocestnosti nejméně na úrovni MMO s jedním cyklem. Použití WBACR AES umožňuje utajit před útočníkem hodnotu klíče IV. Lze použít IOC generované dle 6.1. Před další aktualizační iterací je třeba změnit kódovanou hodnotu K_R z výstupního na vstupního kódování. Pokud je IOC pro K_R použito, útočník nemá v paměti přístupnou jeho otevřenou hodnotu.

6.3 Provázanost IOC

Pro zajištění čerstvosti komunikace je klíčová ochrana hodnoty K_R proti podvržení. Z tohoto důvodu je přítomna pouze v chráněné podobě s aplikovaným IOC. Hodnota K_R je využívána jako argument funkce XOR a její použití i v kódované podobě lze provést při vhodně generovaným IOC základních bloků využívajících WBACR AES (5.2, 6.2) dle 6.1. Hodnota K_R se tak nikdy neobjeví v paměti softwarového agenta v otevřené podobě. Provázanost korespondujících IOC je zachycena na obrázku 4. Generování IOC kompatibilního s více jak jednou operací XOR příliš omezuje počet možných různých kódování a zvyšuje útočníkovu šanci na odhalení předpisu použitého kódování. Změnu kódování z IOC_1 na IOC_2 lze provést pomocí předpočtené tabulky pro operaci identity se vstup-



Obr. 3 Aktualizace klíče relace K_R s využitím WBACR AES

ním kódováním IC_1 a výstupním kódováním OC_2. V případě n-bitového IOC se jedná o tabulku velikosti $2^n * n$ bitů.

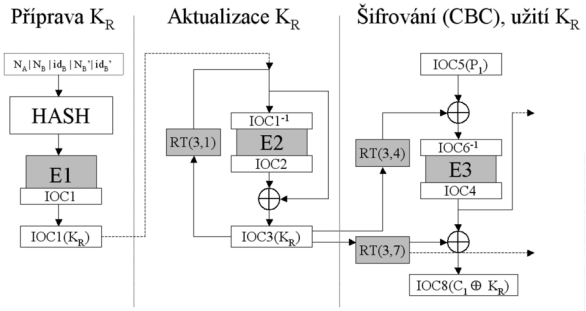
Alternativou k změně kódování z důvodu použitelnosti pro operaci XOR je použití předpočtených tabulek pro tuto operaci s odpovídajícím IOC. V případě dvou n-bitových argumentů se jedná o tabulku velikosti $2^{2n} * n$ bitů. Předpočtené tabulky umožňují nahradit XOR libovolnou jinou „blokovatelnou“ funkcí, pokud by byla vhodnější například z hlediska bezpečnosti.

7 Možnosti využití

Prosazení DRM u koncového uživatele – vlastností ochranného rozhraní, které zajišťují důvěrnost kódu a dat chráněného algoritmu, integritu výkonu a možnost bezpečného řízení prostřednictvím XML licencí, lze využít jako základ DRM architektury ve výpočetním prostředí uživatele. XML parser využívaný pro řízení bezpečnostní proxy je dostupný i pro zpracování příkazů aktualizaci hodnot chráněných algoritmů.

Kontrola využití podpisového klíče umístěného na čipového kartě – pro ochranu před viry, trojskými koni, nebo i samotným uživatelem lze rozšířit ochranu podpisového klíče. I po zadání PINu uživatelem bude moci pouze oprávněný softwarový agent zasílat data určená k podpisu.

Důvěrnost algoritmu/dat – lze použít při potřebě utajit data používaná algoritmem, například hodnot šifrovacích a podpisových klíčů nebo



Obr. 4 Provázanost vstupních a výstupních kódování základních bloků. E1, E2 a E3 označují šifrovací část WBACR AES tabulek pro různé hodnoty klíče. RT(x,y) značí tabulku pro změnu kódování z IOCx na IOCy. P_1 označuje první blok otevřeného textu. C_1 značí první blok zašifrovaného textu

uživatelových privátních informací. Použití kryptografické čipové karty ztěžuje reverzní inženýrství prováděné útočníkem se záměrem odhalit funkčnost poskytovanou algoritmem nebo extrahovat použité návrhové myšlenky.

Omezující podmínkou bránící nasazení ochranného rozhraní může být relativně malá rychlost zpracování požadavku a zaslání odpovědi. Snížení doby odezvy lze dosáhnout použitím výkonnějších čipových karet zrychlující chráněný algoritmus a volbou čipové karty s rychlým hardwarovým akcelerátorem pro algoritmus AES. Dalšího zrychlení lze dosáhnout přechodem na komunikačního rozhraní s vyšší propustností a větší povolenou délkou jednoho příkazu, než poskytují standardní APDU příkazy. Omezující podmínkou je i nutnost fyzické distribuce čipové karty. Díky platformové nezávislosti rozhraní JavaCard lze využít čipových karet distribuovaných za jiným účelem, pokud mají požadované funkční a bezpečnostní vlastnosti. Příkladem mohou být kryptografických čipové karty používané pro digitální podpis nebo SIM karty mobilních telefonů.

8 Závěr

Příspěvek popsal systém pro ochranu vybraných částí kódu s využitím kryptografické čipové karty s podporou JavaCard. Ukazuje, že současné

čipové karty s použitelnou paměti cca 16kB lze využít pro implementaci systému pro řízené využívání kódu více softwarovými agenty, zahrnujícím XML parser pro vzdálenou správu prostřednictvím příkazů ve formátu XML. Dále je popsána implementace autentizačního a transportního protokolu navrženého s využitím principů mobilní kryptografie, který by měl zajišťovat autentizaci, důvěrnost a čerstvost i v případě, že se jedna strana nachází ve výpočetním prostředí kontrolovaném útočníkem. Pro použití v tomto protokolu byl navržen mechanismus generování vstupního a výstupního kódování pro WBACR AES tak, aby ho bylo možno prakticky využít pro šifrovací režim CBC. Dále byla navržena metoda aktualizace a využití klíče relace K_R používaného pro zajištění čerstvosti tak, aby byla chráněna jeho otevřená podoba.

Návrh a implementace ochranného rozhraní byl proveden P. Švendou pod vedením V. Matyáše v rámci diplomové práce s názvem Digital Rights Managment [Sv04]. Oproti původnímu textu je rozšířen návrh využití vstupního a výstupního kódování WBACR AES. Na [Sv04] je dostupná plná verze textu diplomové práce i zdrojové kódy pod GPL licencí.

Literatura

[AES00] *NIST: Advanced Encryption Standard AES*, 2000. Dokument dostupný na URL http://www.nist.com/aes/ (srpen 2003).

[An03] Anderson R.: *'Trusted Computing' FAQ version 1.1.* Dokument dostupný na URL http://www.cl.cam.ac.uk/∼rja14/tcpa-faq.html (srpen 2004).

[Br02] Brandt S.: *Create a quick-and-dirty XML parser.* JavaWorld, 2002. Dokument dostupný na URL http://www.javaworld.com/javatips/jw-javatip128_p.html (srpen 2004).

[Ce02] Červeň P.: *Cracking a jak se proti němu bránit.* Computer-Press, Praha 2002. ISBN 80-7226-382-X

[CEJO02] Chow S., Eisen P., Johnson H., van Oorschot P. C.: *White-Box Cryptography and an AES implementation.* Cloakware Corporation, 2002. Dokument dostupný na URL

http://web.archive.org/web/20040205092333/
http://206.191.60.52/resources/pdf/SAC2002-CW.pdf
(srpen 2004).

[CGJZ01] Chow S., Gu Y., Johnson H., Zakharov, V. A.: *An Aproach to the Obsfucation of Control-Flow of Sequential Computer Programs.* Springer LNCS 2200, Berlin 2001, s. 144–155.

[CTL97] Collberg Ch., Thomborson C., Low D.: *A Taxonomy Of Obsfuscating Transformations.* New Zeland, University Of Aucland, 1997. Dokument dostupný na URL http://www.cs.arizona.edu/~collberg/Research/Publications/CollbergThomborsonLow97a/A4.pdf (srpen 2004).

[CTL98] Collberg Ch., Thombson C., Low D.: *Breaking Abstraction and Unstructuring Data Structures.* University Of Aucland, New Zeland, 1998. Dokument dostupný na URL http://www.cs.arizona.edu/~collberg/Research/Publications/CollbergThomborsonLow97d/A4.ps.gz (srpen 2004).

[DhFe01] Dhem J.–F., Feyt N.: *Present and Future Smart Cards.* Gemplus, France, 2001. Dokument dostupný na URL http://www.itu.dk/courses/DSK/E2002/smart2.pdf (srpen 2004).

[HMST01] Horne B., Matheson L., Sheehan C., Tarjan R.: *Dynamic Self-Checking Techniques for Improved Tamper Resistance.* Springer LNCS 2320, Berlin 2001, s. 141–159.

[Ho98] Hohl F.: *Time Limited Blackbox Security: Protecting Agents From Malicious Hosts.* Springer LNCS 1419, Berlin 1998, s. 92–113.

[ChaAt01] Chang H., Attalah M.: *Protecting Software Code by Guards.* Springer LNCS 2320, Berlin 2001, s. 160–175.

[GaCho01] Gaj K., Chodowiec P.: *Fast Implementation and Fair Comparison of the Final Candidates for AES Using FPGA.* Springer LNCS 2020, Berlin 2001, s. 84–99.

[ISO9798-2] *ISO/IEC 9798-2:1999 Information technology – Security techniques – Entity authentication – Part 2: Mechanisms using symmetric encipherment algorithms.* Popis protokolu je také dostupný v [MOV].

[JC22API] Sun Microsystems, Inc., Palo Alto: *JavaCard 2.2.1 Platform Specification.* 2003. Dokument dostupný na URL http://www.java.sun.com/products/javacard/specs.html (srpen 2004).

[MOV] Menezes A., van Oorschot P., Vanstone S.: *Handbook of Applied Cryptography.* CRC Press 1996/2001. Dostupné také na URL http://www.cacr.math.uwaterloo.ca/hac/ (srpen 2004).

[NCJ01] Nickerson J., Chow S., Johnson H.: *Tamper Resistant Software: Extending Trust In Hostile Environment.* Říjen 2001. Dokument dostupný na URL http://web.archive.org/web/20040205080524/ http://206.191.60.52/resources/pdf/ ACM-01-Trust_in_Hostile_Environments.pdf (srpen 2004).

[NGSCB04] *NGSCB.* Dokument dostupný na URL http://www.microsoft.com/resources/ngscb/default.mspx (srpen 2004).

[OP02] *Open Platform specification.* Dokument dostupný na URL http://www.globalplatform.org/ (srpen 2004).

[RiSch98] Riordan J., Scheider B.: *Environmental Key Generation Towards Clueless Agents.* Springer LNCS 1419, Berlin 1998, s. 15–24.

[RTM01] Rosenblatt B., Trippe B., Mooney S.: *Digital Rights Management: Bussines and Technology.* Indianapolis, Hungry Minds, Inc., listopad 2001. ISBN 0-7645-4889-1

[Ru01] Ruuskanen J.–P.: *JAVACARD.* University of Helsinky, 2001. Dokument dostupný na URL http://www.cs.helsinki.fi/u/campa/teaching/ ruuskanen-final.pdf (srpen 2004).

[SaMo03] Satoh A., Morioka S.: *Hardware-Focused Performance Comparison for the Standard Block Ciphers AES, Camellia, and Triple-DES.* Springer LNCS 2851, Berlin 2003, s. 252–266.

[SaTs98] Sander T., Tschudin Ch.: *Protecting Agents From Malicious Hosts.* Springer LNCS 1419, Berlin 1998, s. 44–60.

[Sv04] Švenda P.: *Digital Rights Managment.* Diplomová práce. Fakulta informatiky, Masarykova universita, Brno 2004. Dokument dostupný na URL http://www.fi.muni.cz/~xsvenda/mst/index.html (srpen 2004).

[TCG04] *Trusted Computing Group.* http://www.trustedcomputinggroup.org/home/ (srpen 2004).

[Za02] Zanero S.: *Smart Card Content Security.* Dipartimento di Elettronica e Informazione, 2002. Dokument dostupný na URL http://www.elet.polimi.it/upload/zanero/papers/scsecurity.pdf (srpen 2004).

[Ze02] Zemánek J.: *Cracking bez tajemství.* ComputerPress, Praha 2002. ISBN 80-7226-703-5