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

XXXII. konference

Sborník příspěvků

Rožmberk nad Vltavou
18.–21. května 2008

Programový výbor

Vladimír Rudolf
Václav Jirovský
Jaroslav Martan
Jan Kasprzak

Sborník příspěvků z XXXII. konference EurOpen.CZ, 18.–21. května 2008

Plzeň 2008. První vydání.
Editoři: Vladimír Rudolf, Jiří Sitera
Sazba a grafická úprava: Ing. Miloš Brejcha – Vydavatelský servis, Plzeň
e-mail: servis@vydavatelskyservis.cz

Tisk: TYPOS, Tiskařské závody, a. s.
Podnikatelská 1 160/14, Plzeň

Příspěvky neprošly redakční ani jazykovou úpravou.

ISBN 978-80-86583-14-3

Obsah

Taxonomie kybernetických hrozeb

Václav Jirovský

E-mail: jirovsky@dkm.cz

Pod slovem „taxonomie“ obvykle rozumíme nějakou uspořádanou klasifikaci nebo kategorizaci. Při tomto uspořádáni se používají některé významné společné znaky nebo vlastnosti, které určují příslušnost příslušného objektu do dané třídy. Vytvoření taxonomického systému je důležité zejména tam, kde je potřeba objekty nějakého systému dále hodnotit. To je i případ nových fenoménů kybernetických hrozeb jako jsou kyberterorismus nebo kybernetická kriminalita.

Přesto, že existuje celá řada mezinárodních dohod, deklarací a stanovisek, jednotné třídění v této oblasti se neustálilo. V následujícím příspěvku se pokusíme o taxonomický přístup vycházející ze tří hledisek:

- podle účastníků, efektu a cílů
  - populace internetu,
  - druhu zásahu cíle,
  - účastníků hrozby,
- podle charakteru hrozby dělené na
  - základní hrozby,
  - aktivační hrozby,
  - podkladové hrozby
- podle technologických vlastností nebo nejvhodnějšího síťového modelu
  - umístění v OSI/ISO architektuře
  - interaktivita, etc.

Populace kyberprostoru je reprezentována virtuálními osobnostmi, které jsou projekcí reálné osobnosti člověka do kyberprostoru. Tyto populace vytvářejí virtuální komunity, které je možno chápat jako globální seskupení virtuálních osobností bez omezení hranicemi států, svázaných společnými myšlenkami, vírou, politickým názorem, zkušenostmi, zájmem etc. Zde můžeme pozorovat i častou politickou nebo náboženskou motivaci. Mezi zajímavé objekty kyberprostoru patří virtuální korporace tvořená subjekty zamřenými na stejný tržní segment.

Budeme-li se zabývat pouze hlediskem kyberterorismu, pak je možné od sebe odlišit politický terorismus, separatismus a další druhy terorismu projevující se na internetu (např. sektářský terorismus, revoluční terorismus, a další).

Otázka kdo jsou útočníci je namístě pokud sledujeme zájmy útočícího. Může se jednat o hackerské skupiny najaté pro uskutečnění útoku (H4H), zvláštní jednotky teroristických buněk, zvláštní vojenské jednotky apod. Typickým znakem těchto uskupení je vysoká kvalita útoků, specifické strategie a specifické cíle. Zcela jiná je charakteristika ideologických sympatizantů nebo hledačů vzrušení. Vedle obecného motivu, kterým je většinou exhibicionismus, je významný počet útoků ovlivňován politicky.

Existence tzv. nepřímého kyberterorismu byla donedávna opomíjena, i když jeho latentní chování je mnohem nebezpečnější než některé přímé kybernetické útoky. DFo této skupiny můžeme zařadit terorismus spojený s IT bez přímého vztahu k existující IT infrastruktuře a je příznačně svázán s vývojem „informačního věku“, informatikou a telekomunikacemi. Je obecně založen na využití pocit svobody podporované vnímáním volnosti na internetu, obecně nízké úrovni úsudku o kvalitě předávané informace a používá sítě IT jako nástroj. Do této skupiny patří např. mediální terorismus, hacktivismus, zneužití procesů a výkonu IT, procesní terorismus a IT governance.

Při taxonomické struktuře založené na charakteru hrozby je nutno oddělit základní hrozby od hrozeb, které jsou pro realizaci útoku nezbytné od hrozeb podkladových a aktivačních. Přitom vzájemná závislost hrozeb podmiňuje jejich uskutečnění a při vynechání některého z článků realizace hrozby selže.

Taxonomie založená na síťovém modelu vychází z klasifikací události na bázi vrstevnatého modelu OSI/ISO. Kvalifikace hrozby na jednotlivých vrstvách tvoří ucelenou strukturu pro popis kybernetické hrozby.

Uvedené taxonomie používají různé pohledy na kybernetickou hrozbu. Zatímco dělení podle účastníků, efektu a cílů popisuje spíše sociální a psychologický pohled na kybernetickou hrozbu a je vhodná pro vytvoření profilu útočníka, taxonomie podle charakteru hrozby popisuje spíše technologii útoku a je výhodná pro nastavení bezpečnostní politiky. Poslední uvedená taxonomie založená na síťovém modelu popisuje architektonický pohled na útok a bude užitečná pro nastavení IDS systémů nebo podobných opatření. Je však nutno uvést, že existují i další taxonomie a cesta k jejich sjednocení nebude jednoduchá.

Češi, Internet, bezpečnost

Petr Soukup

E-mail: soukup@fsv.cuni.cz

Abstrakt

Cílem příspěvku je nabídnout bližší pohled na užívání Internetu v ČR a srovnat užívání Internetu u nás se zahraničím. Na základě výsledků kvantitativních šetření české populace (osob starších 11 let) World Internet Project 2005–2007 (Fakulta sociálních studií MU) je provedena typologie uživatelů a poukázáno na specifické rysy užívání Internetu u jednotlivých skupin. Pozornost je věnována též bezpečnostním aspektům užívání Internetu a vnímání bezpečnosti na Internetu samotnými uživateli. Pro doplnění obrazu nejmladších uživatelů Internetu je využito šetření Pocit bezpečí provedeného Institutem sociologických studií Fakulty sociálních věd UK v lednu 2008. V tomto šetření byla u 14–16 letých žáků a studentů v ČR zjišťována míra obav z kriminality včetně některých deliktů v kyberprostoru (viry, hacking, phishing), dále míra viktimizace a podílu na této trestné činnosti. Doplňkově bylo též sledováno, zda mladí považují určitá jednání ze trestné (viry, phishing, hacking, sdílení souborů) a zda by v případě, že by o této činnosti věděli, nahlásili činnost Policii ČR.

Úvod

Internet je fenomén, bez kterého si dnešní člověk neumí přestavit život. Tato věta by mohla být určitě mottem mnoha lidí, ale rozhodně ne všech. Cílem tohoto textu je zejména ukázat na rozdíly mezi uživateli a neuživateli Internetu v České republice a potažmo zpochybnit výše uvedené motto. Pro plastičtější obraz je popsána i vnitřní struktura uživatelů Internetu v České republice, protože užívání Internetu nelze pouze jako chápat jako rozhraničení mezi uživateli a neuživateli (tzv. digital divide), ale spíš jako širokou škálu užívání. K tomuto

Tento text vznikl díky projektu bezpečnostního výzkumu„Problematika kybernetických hrozeb z hlediska bezpečnostních zájmů České republiky“ – VD20072010B01. Částečná podpora byla též poskytnuta z výzkumného záměru FSV UK „Rozvoj české společnosti v EU: výzvy a rizika“ (MSM 0021620841).

účelu budeme využívat dat z velkých kvantitativních šetření prováděných akademickými institucemi v ČR v posledních letech, zejména šetření ISSP 2007[1] a WIP (2005, 2006 a 2007)[2]. Kromě užívání Internetu a různých jeho služeb bude pozornost orientována na vnímání bezpečnosti na Internetu. Doplňkově také bude využito šetření Pocit bezpečí[3]. V tomto šetření byla u 14–16 letých žáků a studentů v ČR zjišťována míra obav z kriminality včetně některých deliktů v kyberprostoru (viry, hacking, phishing), dále míra viktimizace a podílu na této trestné činnosti. Cílem do budoucna je zmapování bezpečnostních rizik, které hrozí české internetové populaci ve vlastním výzkumu, který bude součástí šetření WIP 2008.

Na počátku je důležité upozornit čtenáře, že cílem tohoto textu není prověřování velkých sociologických teorií, ale empirický popis české populace. Proto také nejsou žádné teorie rozebírány v úvodu, nevěnujeme pozornost koncepci digital divide, ani jeho kritikám.

Češi a volný čas

Začneme trošku netradičně a zeširoka a pokusíme se podívat na to co dospělé Češky a dospělí Češi dělají ve svém volném čase[4]. Toto odbočení není zcela samoúčelné, pomůže nám totiž pochopit užívání vs. neužívání Internetu v širším rámci. V rámci šetření ISSP 2007 zaměřeného na volný čas (výzkum provedený Sociologickým ústavem AV ČR[5] resp. pro něj agenturou GFK) byla lidem na počátku dotazníku nabídnuta paleta činností, které mohou dělat ve svém volném času a u každé nabízené mohli odpovědět, že ji provozují *denně, několikrát týdně, několikrát za měsíc, několikrát za rok nebo méně často* a případně *nikdy*.

Graf 1 ukazuje výsledky pro českou dospělou populaci pro vybrané (nejčastější) činnosti ve volném čase. Zřejmě nikoho nepřekvapí, že nejčastěji provozovanou činností je sledování pasivních médií (TV, DVD, video sleduje denně téměř 70 % populace[6], naopak 1 % nikdy). Další velmi častou činností je poslech

[1] Bližší informace o tomto šetření lze nalézt na http://samba.fsv.cuni.cz/∼soukup/vyzkumy, dotazník lze nalézt na tamtéž.

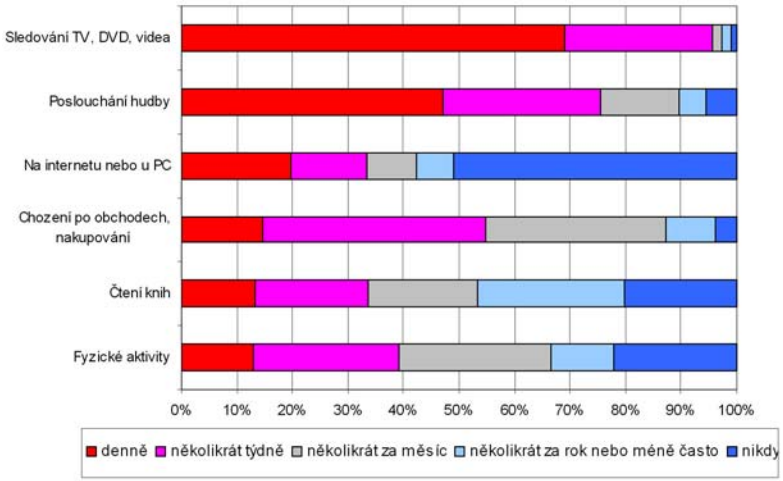
[2] Bližší informace o tomto šetření lze opět nalézt na http://samba.fsv.cuni.cz/∼soukup/vyzkumy.

[3] Šetření provedené Institutem sociologických studií Fakulty sociálních věd UK v lednu 2008 na 14–16 letých žácích, dotazník je dostupný na http://samba.fsv.cuni.cz/∼soukup/vyzkumy.

[4] Jsme si vědomi, že redukce na dospělou populaci je s ohledem na sledovaný fenomén (Internet) značně nevhodná, Bohužel šetření ISSP 2007, ze kterého zde vycházíme, bylo provedeno jen na dospělé populaci a zjištění pro mladší populaci nejsou dostupná. V dalších částech textu budou díky výzkumům zmapovány i děti a mládež cca od 12 let věku.

[5] Děkuji doc. Tučkovi ze Sociologického ústavu za možnost využít tato data již v průběhu jeho grantového projektu.

[6] Dodejme, že u dětí(10–14 let), se podíl těch co sledují TV denně (studie Jak čtou české děti, 2002, Gabal Consulting) pohybuje okolo 87 %.



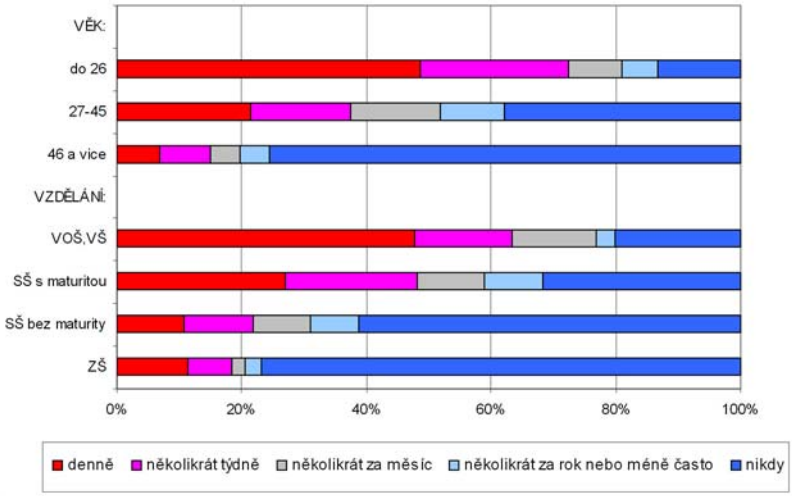
Zdroj: ISSP 2007, n = 1 222

Graf 1: Četnost činností prováděných ve volném čase (dospělá populace v ČR)

hudby (necelá polovina populace denně, 6 % nikdy). Na třetím místě pomyslného žebříčku je čas trávený na Internetu a u počítače (pětina populace každý den, ale celá polovina nikdy!). Zde je patrný první náznak rozdělení na uživatele a neuživatele Internetu, zhruba polovina se řadí mezi neuživatele, dodejme zde specificky ve svém volném čase. Nicméně je známo, že ten, kdo užívá PC a Internet v zaměstnání či škole užívá je zpravidla i ve svém volném čase a naopak. Užívání Internetu ve volném čase brát jako indikátor užívání celkového. Nabízí se tedy otázka: Kdo tedy patří mezi častější uživatele Internetu ve svém volném čase a kdo naopak užívá méně často? Podíváme se jen na základní sociodemografické charakteristiky (věk, vzdělání, pohlaví, velikost obce a sebezařazení osob na společenském žebříčku).

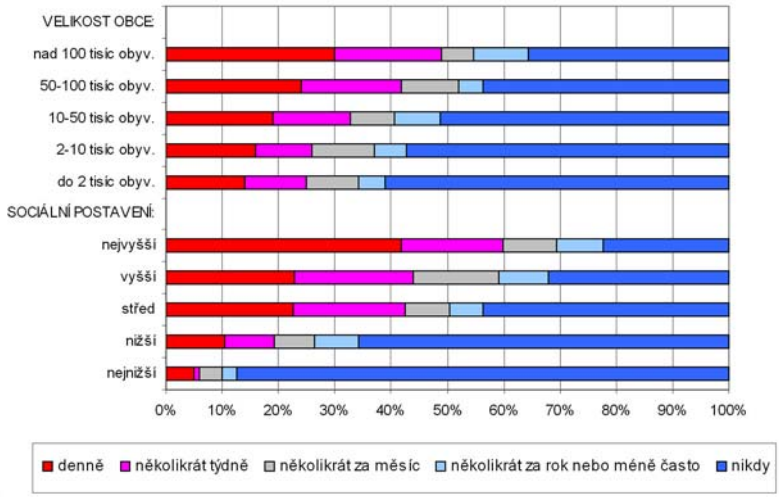
Z grafu 2 je patrné, že klíčovou charakteristikou ovlivňující užívání vs. neužívání Internetu a také frekvenci užívání je věk. Zatímco z nejmladších dospělých (do 26 let[7]) se věnuje ve volném čase počítačům a Internetu denně téměř polovina (naopak nikdy jen 13 %), u osob starších 46 let se věnuje ve volném čase počítačům denně 7 % a nikdy tři čtvrtiny. Rozdíly jsou i mezi vzdělanostními kategoriemi (polovina absolventů VŠ či VOŠ pracuje s počítačem ve volném čase denně, u osob se základním vzděláním pouze zhruba desetina) a také mezi muži

[7]Tato hranice byla zvolena jako hranice, kdy už bude mít většina lidí ukončené studium a proto zahrnuje mladé pracující a mladě studující, další kategorie již potom převážně pracující, nezaměstnané, osoby na rodičovské dovolené a důchodce.



Zdroj: ISSP 2007, n = 1 222

Graf 2: Četnost trávení volného času u počítače a na Internetu (dospělá populace v ČR) dle věku a vzdělání



Zdroj: ISSP 2007, n = 1 222

Graf 3: Četnost trávení volného času u počítače a na Internetu (dospělá populace v ČR) dle velikosti bydliště a sociálního postavení

a ženami (častěji tráví s počítačem volný čas muži než ženy[8]). Z grafu 3 vidíme, že rozdíly (částečně ovšem vysvětlitelné odlišnou vzdělanostní a věkovou strukturou) lze nalézt i mezi obyvateli různě velkých obcí. Zatímco v malých obcích (do 2 tisíc obyvatel) se věnuje počítači a Internetu ve volném čase denně 14 % osob (naopak nikdy 61 %), ve velkoměstech (nad 100 tisíc obyvatel) jsou tyto podíly téměř vyrovnány (30 % denně resp. 36 % nikdy). Obdobně nalezneme rozdíly mezi osobami z různých společenských vrstev. Zatímco lidé, kteří jsou podle svých slov na nejnižších stupních společenské hierarchie volný čas s počítačem a Internetem běžně netráví (5 % denně, 87 % nikdy), lidé z horních vrstev naopak svůj volný čas tráví takto velmi často (42 % denně, resp. 22 % nikdy).

Shrneme-li uvedená zjištění o volném čase dospělé české populace, platí, že počítače a Internet jsou poměrně oblíbeným koníčkem, zejména pro osoby mladší, vzdělanější, spíše muže, z větších měst. Ostatně tato charakteristika je platná i pro uživatele Internetu vůbec to nejen dnes, ale i v dobách dřívějších. Na základě výzkumu Internetové populace v ČR v roce 2000, bylo zjištěno, že oproti obecné populaci výrazně dominují muži, mladší osoby a lidé z velkoměst (Soukup 2001).

Češi a Internet

Po exkurzu o volném čase věnujme pozornost již jen užívání Internetu a jeho specifikům v České republice v posledních letech. Budeme vycházet z výzkumů WIP (World Internet Project) 2005, 2006 a 2007 provedených na FSS MU[9] (resp. reálný sběr dat prováděla ve všech letech agentura STEM). Tento výzkum se soustředí tématicky na Internet a na populaci od 12 let, a proto lze tato data považovat za velice vhodná pro účely tohoto textu.

Ještě než využijeme jen tato data, zaměřme se na několik čísel ze statistických přehledů. Velikost populace ČR zveřejňuje ČSÚ, poslední údaj (k 1. 10. 2007) hovoří o 10349372 osob. Další zajímavou hodnotou je počet uživatelů Internetu, dle šetření NetMonitor je aktuální hodnota (únor 2008) 4 635 734. Při uvažování populace od 12 let (cca 9,3 milionu osob) lze tedy odhadovat penetraci Internetu (podíl uživatelů) okolo 50 %. Tedy z osob starších 12 let v ČR zhruba 4,6 milionu osob neužívá Internet. Nutno ovšem dodat, že pokud budeme chtít širší pohled na tzv. kyberprostor, pak musíme zohlednit ještě užívání mobilního telefonu. Z výzkumu WIP 2007 plyne, že z osob neužívajících Internet pouze 18 % nemá mobilní telefon a lze tedy uzavřít, že český kyberprostor zahrnuje cca 8,5 milionu osob starších 11 let.

[8] Samozřejmě, že teď je prostor na spekulace, proč tomu tak je. Jednoduché vysvětlení většího podílu žen na domácích pracích se nabízí téměř automaticky, nicméně šetření ISSP 2007 takovou analýzu neumožňuje.

[9] Děkuji dr. Šmahelovi z FSS MU za možnost využívat tato data.

Zaměřme se teď opět pouze na uživatele Internetu, nejdříve lehce ve vývoji. Zatímco v roce 2005 byl naměřen podíl uživatelů Internetu (u osob starších 11 let) ve výši 40 %, v roce 2007 je to již 55 %. To je stále poměrně značný nárůst, i když samozřejmě ve srovnání s podílem osob užívajících mobilní telefon (cca 90 % osob starších 11 let) jde o hodnotu poměrně malou.

Typologie českých uživatelů Internetu

Skupina uživatelů Internetu je ale poměrně nestejnorodá a pro účely dalšího popisu jsme provedli typologizaci. Zkoušením nejrůznějších postupů a také četbou zahraniční literatury jsme dospěli k názoru, že vhodná typologie je založena na délce užívání Internetu (v letech) a počtu hodin na Internetu z domova týdně. Na základě těchto dvou dimenzí bylo vytvořeno pět typů uživatelů Internetu v České republice. Tyto typy, resp. jejich podstatu vystihuje schéma 1[10].

Schéma 1: Typologie uživatelů Internetu v ČR

Tabulka 1: Podíly jednotlivých typů Internetových uživatelů v ČR (Internetová populace starší 11 let v ČR) a odhady velikosti v populaci

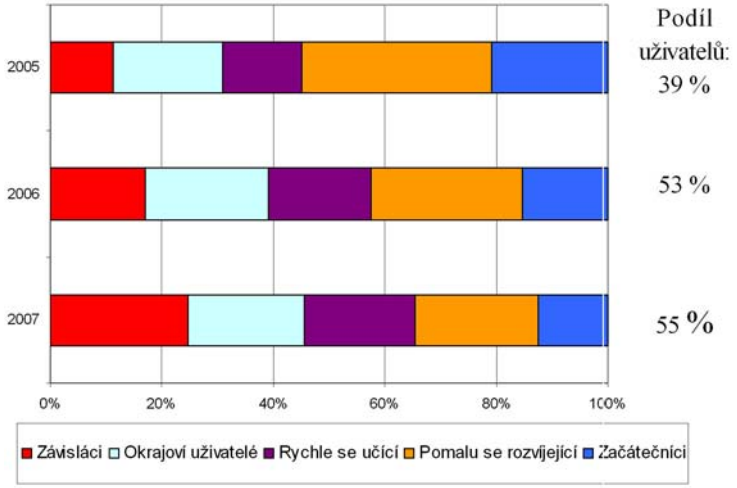
Typ	**Podíl z uživatelů Internetu**	**Odhad počtu v ČR (tis.)**
Začátečníci	13 %	577
Pomalu se rozvíjející	22 %	1 026
Rychle se učící	20 %	914
Okrajoví uživatelé	21 %	973
Závisláci	25 %	1 144

Zdroj: WIP 2005–2007, n = 2 378
Údaje o podílech uživatelů jsou sloupcová procenta

[10] Označení typů je nutno brát jako snahu vystihnout jejich podstatu. Za případné náměty na vylepšení názvů bude autor textu vděčný.

Důležité je zaměřit se na strukturu české Internetové populace, tedy jaké jsou podíly jednotlivých typů. Tyto proporce a odhady počtů jednotlivých typů uživatelů v populaci[11] jsou uvedeny v tabulce 1.

Pro doplnění dynamiky rozvoje Internetu v ČR ještě uvádíme graf popisující vnitřní strukturu uživatelů Internetu v letech 2005–2007 (graf 4).



Zdroj: WIP 2005–2007, n = 2 378

Graf 4: Vývoj podílu jednotlivých typů uživatelů Internetu v ČR v letech 2005–2007 (Internetová populace starší 11 let v ČR)

Z grafu 4 je zřejmé, že nejen narůstá podíl uživatelů Internetu, ale i jejich struktura se proměňuje a logicky přibývá osob, které užívají Internet více let a také tráví na Internetu více času (nejmarkantnější je nárůst podílu typu pracovně označeného jako závisláci (tj. lidé, kteří tráví na Internetu z domova alespoň 5 hodin týdně a užívají Internet 5 a více let). Samozřejmě ani tato skupina (v populaci cca 1,1 milionu) není homogenní. Pracovně jsme si ji ještě rozdělili na 4 podskupiny dle doby strávené na Internetu z domova (viz tabulka 2).

Z této tabulky plyne, že v populaci je cca 100 tisíc zkušených uživatelů Internetu, kteří tráví s tímto médiem alespoň 30 hodin týdně doma a zhruba dvojnásobek těch, co tráví takto alespoň 20 hodin. Tyto odhady[12] jsou velmi důležité s ohledem na projekt, který řešíme, neboť tyto osoby jsou potenciálními pachateli (resp. i oběťmi) počítačové kriminality.

[11] S ohledem na výzkum WIP rozuměj v populaci osob starších 11 let.

[12] Odhady nepovažujeme za zcela spolehlivé. Důvodem je jednak nespolehlivost dotazníkových šetření obecně a také skutečnost, že takových respondentů bylo ve výzkumu málo, a proto je nutno být při práci s uvedenými hodnotami obezřetný.

Tabulka 2: Detailnější rozdělení typu „závisláci“ (starší 11 let v ČR) a odhady velikosti v populaci

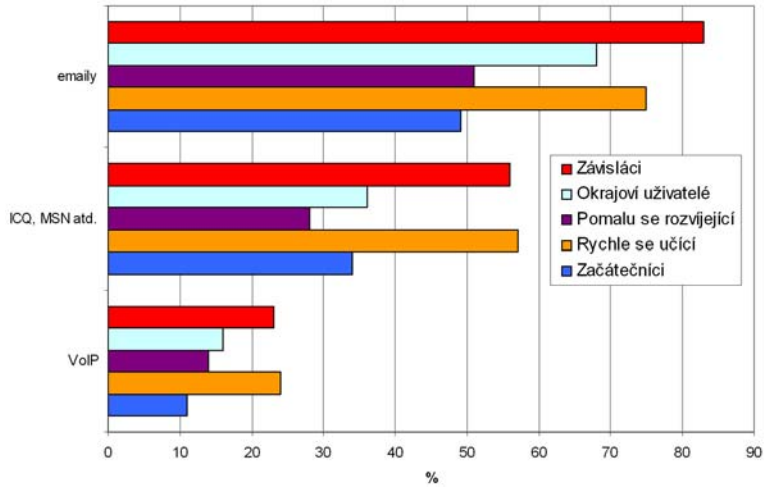
Na Internetu z domova	Podíl ze závisláků	Odhad počtu v ČR (tis.)
5–10 hodin	44 %	500
11–20 hodin	37 %	430
21–30	19 %	210
Více než 30 hodin	8 %	90

Zdroj: WIP 2007, n = 437
Údaje o podílech závisláků jsou sloupcová procenta

Užívání služeb na Internetu

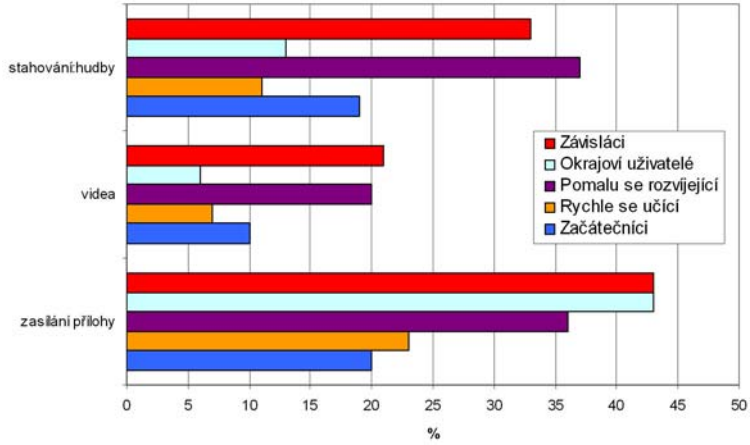
Nyní stručně zaměříme pozornost na rozdíly mezi uvedenými typy uživatelů Cílem této části je zjistit, zda se jednotlivé typy liší v užívání různých služeb na Internetu, zda mají stejné obavy z Internetu a jaké jsou sociodemografické odlišnosti těchto typů.

Nejdříve zaměříme pozornost k užívání jednotlivých služeb na Internetu (viz grafy 5, 6 a 7).



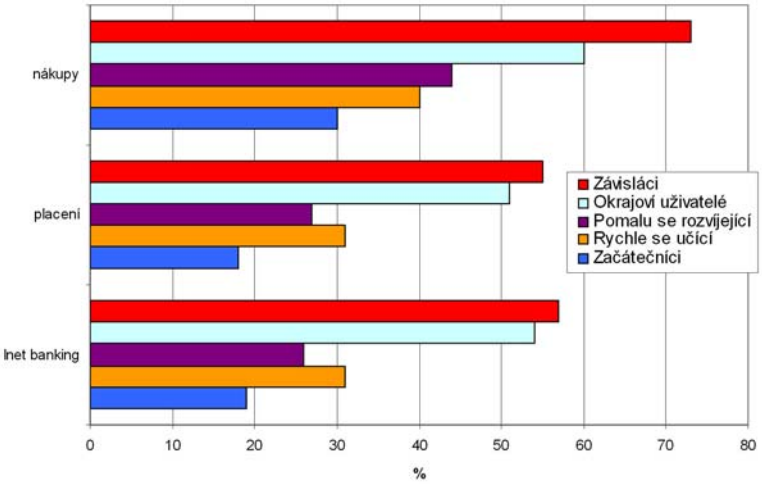
Zdroj: WIP 2005–2007, n = 2 378

Graf 5: Využívání komunikačních služeb na Internetu **alespoň jedenkrát denně** u jednotlivých typů uživatelů Internetu v ČR v letech 2005–2007 (Internetová populace starší 11 let v ČR)



Zdroj: WIP 2005–2007, n = 2 378

Graf 6: Využívání operací se soubory na Internetu **alespoň jedenkrát denně** u jednotlivých typů uživatelů Internetu v ČR v letech 2005–2007 (Internetová populace starší 11 let v ČR)



Graf 7: Využívání služeb obchodní povahy na Internetu **alespoň někdy** u jednotlivých typů uživatelů Internetu v ČR v letech 2005–2007 (Internetová populace starší 11 let v ČR)

Obecně platí, že čím déle lidé užívají Internet a čím více času na něm tráví, tím více služeb využívají. Ze závisláků používá email denně více než 80 %, u začátečníků a pomalu se rozvíjejících je to jen polovina. Z grafů je patrné, že lidé trávící na Internetu nejvíce času (závisláci a rychle se učící) používají nejvíce ze všech komunikační služby (email, komunikátory a VoIP). Lidé, kteří užívají Internet dlouho (závisláci a pomalu se rozvíjející) jsou častějšími uživateli složitějších služeb (stahování, zasílání příloh). Závisláci používají nejčastěji ze všech doprovodné služby obchodního charakteru (73 % alespoň někdy nakupuje, 55 % alespoň někdy přes Internet platí a 57 % užívá Internet banking), což je oproti začátečníkům zhruba 2 až 3krát více. Platí tedy obecná teze, kterou můžeme vyjádřit českým příslovím: s jídlem roste chuť (rozuměj čím déle a častěji Internet užívám, tím více jeho služeb užívám), ovšem s tím dodatkem, že některé služby jsou spíše pro dlouhodobé uživatele a některé spíše pro časté uživatele (nepřímo to zohledňuje dělení na nejmladší uživatele, kteří užívají Internet intenzivně, ale vzhledem k věku krátce a o něco starší uživatele, kteří využívají služby delší dobu, ale často ne tak intenzivně[13]).

Obavy na Internetu

Pro indikaci různé míry obav na Internetu můžeme využít jediné otázky ze šetření WIP a to sice obavy z placení kartou na Internetu. Nejdříve se podívejme na vývoj těchto obav v letech 2005–2007 (graf 8). Z grafu 8 jasně vidíme, že roste podíl osob, které mají kreditní kartu (nárůst zejména mezi rokem 2005 a 2006) a naopak klesá podíl osob, které se placení obávají. Z hlediska výše uvedené typologie platí, že nejvýraznější obavy mají začátečníci, naopak nejmenší závisláci (viz graf 9). V opačném poměru je rozšíření kreditních karet, 20 % závisláků nemá kartu, ze začátečníků 36 %.

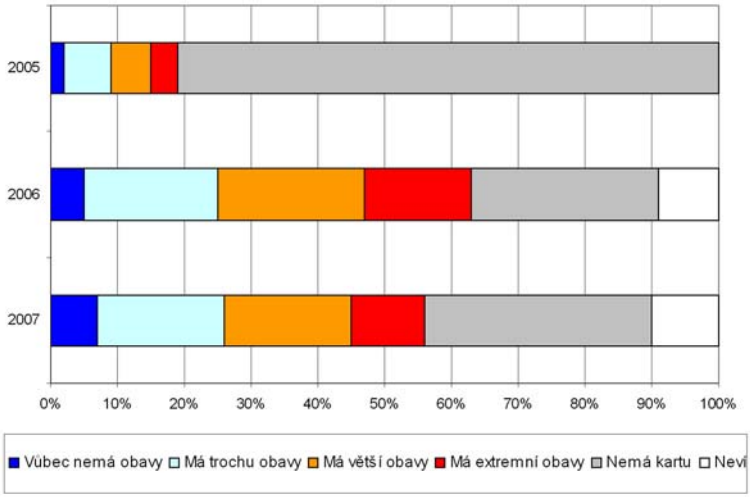
Lze tedy rozšířit předchozí poznatky v tom smyslu, že délka a frekvence užívání Internetu ovlivňuje nárůst pocitu bezpečí na Internetu.

Sociodemografické charakteristiky jednotlivých typů

Ve stručnosti se ještě zaměřme na sociodemografickou odlišnost (věk, vzdělání, pohlaví a velikost obce) jednotlivých typů uživatelů a porovnejme je navíc s neuživateli (tabulka 3).

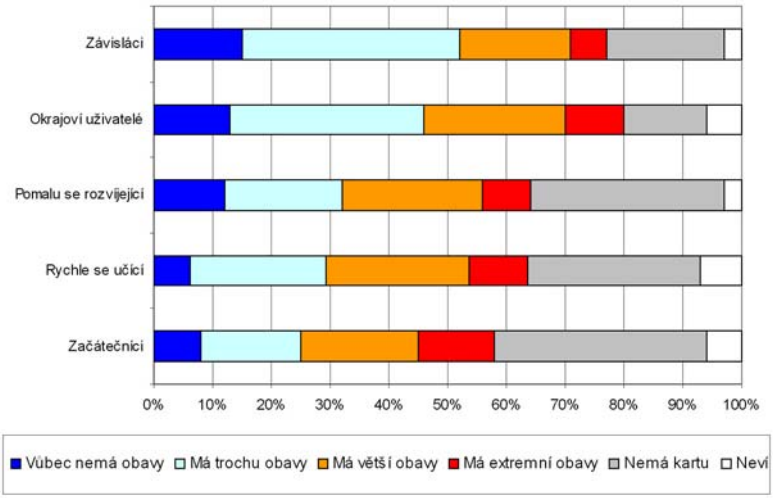
Prvním výrazným rozdílem je průměrné stáří, zatímco u jednotlivých typů uživatelů osciluje okolo 30–40 let, u neuživatelů je to výrazně více cca 51 let.

[13] Samozřejmě tato argumentace by zasloužila detailnější analýzu, částečné potvrzení uvádíme dále v textu v části věnované sociodemografickému profilu jednotlivých typů.



Zdroj: WIP 2005–2007, n = 4 807

Graf 8: Vývoj obav z užívání karet na Internetu v letech 2005–2007 (populace starší 11 let v ČR)



Zdroj: WIP 2007, n = 868

Graf 9: Obavy z užívání karet na Internetu u jednotlivých typů uživatelů Internetu v ČR v roce 2007 (Internetová populace starší 11 let v ČR)

Tabulka 3: Sociodemografické profily neuživatelů Internetu a jednotlivých typů uživatelů v ČR (populace starší 11 let v ČR)

		Neuži-vatelé	Uživatelé				
			Začátečníci	*Rychle se učící*	*Pomalu se rozvíjející*	*Okrajoví uživatelé*	*Závisláci*
věk	průměr	51,16	31,12	32,92	28,47	38,56	32,81
vzdělání	základní	26	40	31	43	10	20
	SŠ bez maturity	50	29	25	23	17	18
	SŠ bez maturity	18	28	36	27	48	38
	VŠ, VOŠ	5	4	8	6	26	24
pohlaví	muž	44	49	48	57	47	64
	žena	56	51	52	43	53	36
počet obyvatel	Do tisíce	17	17	17	15	11	13
	1–2 tis.	10	10	9	8	10	8
	2–5 tis.	11	12	11	9	11	10
	5–20 tis.	18	19	17	17	18	17
	20–90 tis.	21	17	22	25	17	23
	Nad 90 tis.	24	26	23	26	32	30

Zdroj: WIP 2005–2007, n = 4 807
Údaje s výjimkou průměrného věku jsou sloupcová procenta

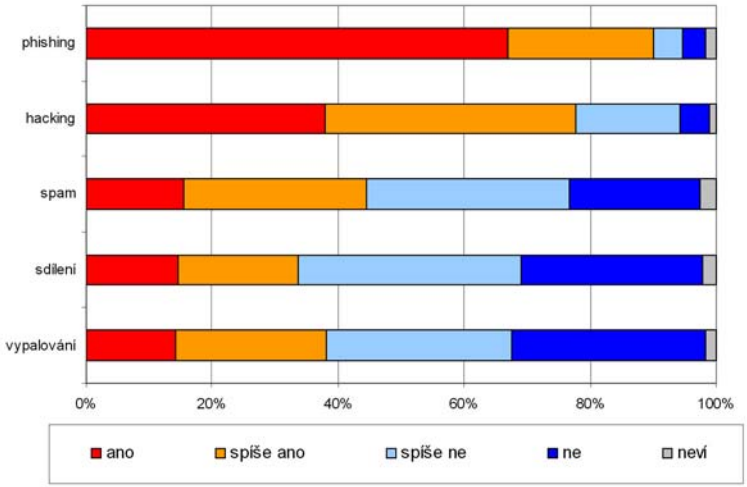
Výrazné rozdíly najdeme i vzdělanostních strukturách, zatímco mezi neuživateli dosáhlo vyššího než maturitního vzdělání jen 5 %, u typů, které jsme si pracovně nazvali okrajoví uživatelé a závisláci je to více než čtvrtina (26 % resp. 24 %). Také struktura z hlediska pohlaví je odlišná dle užívání/neužívání Internetu, resp. dle míry užívání. Zatímco mezi neuživateli převažují ženy (56 %), mezi uživateli-závisláky naopak výrazně převažují muži (64 %). Z hlediska velikosti obce se potvrzuje tendence naznačená již na počátku při analýze užití volného času, tedy více uživatelů Internetu je ve větších obcích a vice versa.

Čeští teenageři a Internet – naděje do budoucna?

Pro doplnění obrazu české Internetové populace uvedeme některé výsledky z šetření Pocit bezpečí provedeného Institutem sociologických studií Fakulty sociálních věd UK v lednu roku 2008. V tomto šetření byla u zhruba 800 žáků a studentů v ČR (14–16 letých) zjišťována míra obav z kriminality včetně některých deliktů v kyberprostoru (viry, hacking, phishing), dále míra viktimizace a podílu na této trestné činnosti. Doplňkově bylo též sledováno, zda mladí považují určitá jednání ze trestné (viry, phishing, hacking, sdílení souborů) a zda by v případě, že by o této činnosti věděli, nahlásili činnost Policii ČR.

Trestnost a obavy

První oblastí, na níž byli náctiletí dotazováni, bylo posouzení trestnosti jednotlivých činů. Z grafu 10 je zřejmé, že největší proporce žáků a studentů (cca 90 %) považuje za trestný phishing, následovaný hackingem, naopak sdílení a vypalování hudby a videa spíše za trestné nepovažují (jen necelých 40 %).



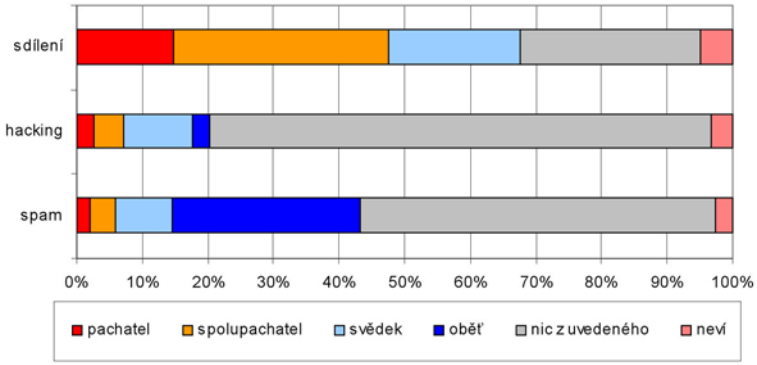
Zdroj: Pocit bezpečí, n = 689

Graf 10: Trestnost činů v kybersprostoru (14–16 letí v ČR)

U tří typů útoků na Internetu (viry & spyware, phishing a hacking) byli náctiletí vyzváni, aby uvedli, zda se jich obávají. Největší obvavy mají dle svých slov z virů a spyware (40 % se obává velice nebo aspoň trochu), naopak obavy z phishingu a hackingu jsou spíše sporadické (26 % resp. 19 %). Důvod, proč se mladí bojí spíše virů než phishingu a hackingu plyne zřejmě z reálné zkušenosti s těmito typy útoků. Zatímco viry či spyware zažívá téměř každý uživatel, hacking je zatím spíše neznámý a phishing byl v době šetření (leden 2008) také spíše vzácným jevem[14]).

Pro doplnění obrazu české mládeže jsme zjišťovali podíl na různých deliktech (sdílení, hackingu rozesílání spamu). Výsledky můžeme číst v grafu 11. Nejčastější je sdílení hudebních souborů a videí (téměř polovina náctiletých uvádí, že tuto činost provádějí nebo se na ní podílejí a další pětina ví o sdílejících ka-

[14]Samozřejmě po útocích na Českou spořitelnu lze očekávat větší citlivost populace k těmto útokům a také zvýšení obav. Zda se tato předpověď potvrdí, zjistíme v plánovaném šetření v září 2008.



Zdroj: Pocit bezpečí, n = 689

Graf 11: Účast na jednotlivých jednáních v kybersprostoru (14–16 letí v ČR)

marádech), naopak hacking a rozesílání spamu je řídké (zhruba 5 % oslovených uvedlo podíl na těchto deliktech[15]).

Závěrem ještě byli náctiletí tázáni, zda by v případě, že by se stali svědky určitých deliktů, nahlásili toto Policii ČR. V případě phishingu projevilo ochotu nahlásit věc Policii zhruba dvě třetiny náctiletých, u hackingu 40 % a v případě sdílení souborů 5 %[16].

Stručné shrnutí

Zkusíme-li stručně shrnout poznatky o české Internetové populaci, lze konstatovat následující. Internet užívá o něco více než polovina populace (starší 11 let, pro mladší neexistují seriózní odhady). Internet užívají výrazněji lidé mladší, vzdělanější a z větších měst. Internetová populace není jednolitá, pro účely našich analýz jsme ji rozdělili pracovně na pět typů (dle délky užívání v letech a doby strávené na Internetu z domova týdně). Jednotlivé typy uživatelů se mezi sebou výrazně liší v typu užívaných služeb a také v obavách z Internetu (platí, že ti co užívají Internet nejintenzivněji a nejvíce služeb mají obavy nejnižší). Doplňkově jsme věnovali pozornost dospívajícím a jejich náhledu na kriminogenní faktory na Internetu. Z provedeného šetření je patrné, že některá jednání (sdí-

[15] Obdobnou proporci podílu na hackingu zjistilo i mezinárodní reprezentativní šetření české mládeže ISRD (Filozofická fakulta UK, 2006, n = 2 283). Děkuji tímto vedoucímu projektu doc. Buriánkovi za možnost využít jejich výzkum. Obecně lze tato čísla považovat za mírně nadhodnocená oproti realitě a navíc nediferencující jednotlivé formy hackingu.

[16] Dodejme, že nízká míra nahlašování deliktů Policii není u mládeže obecná, v případě vražd uvedlo 96 % náctiletých, že by delikt nahlásilo.

lení, spam) nepovažují mladí většinou za trestná a často tyto aktivity provozují (zejména sdílení dat). Obavy mají mladiství zejména z útoku virů či spyware při srovnání s phishingem a hackingem. Nezbývá než dodat, že na podzim roku 2008 plánujeme velké šetření, které by mělo zmapovat oblast obav, z Internetu, bezpečnostních incidentů, provozování různých aktivit na hraně zákona. Po provedení tohoto šetření chceme výsledky konzultovat s experty na danou problematiku a vyvodit důsledky zejména v oblasti trestněprávní politiky. Doplňkově budeme vést rozhovory i se specifickými skupinami obyvatel (mladí, neuživatelé), abychom detailněji porozuměli zjištěným skutečnostem z velkých reprezentativních šetření.

Literatura

[1] Soukup, P. *Kdo používá Internet v České republice?* SDA Info, 2001, vol. 3, No. 2, s. 1–2.

Úmluva o počítačové kriminalitě

Tomáš Gřivna

E-mail: Tomas.Grivna@seznam.cz

1 Úvodem

Převratný vývoj moderních informačních a telekomunikačních technologií ovlivňuje všechny stránky lidského života. Na jedné straně usnadňuje a racionalizuje nesčetnou řadu lidských činností, na druhou stranu integrace telekomunikačních a informačních systémů skýtá možnosti pro páchání trestné činnosti. Připojením na komunikační a informační služby vytváří jednotliví uživatelé určitý druh společného prostoru, který lze nazvat „kyberprostorem". Zároveň se tak vzniká jako nechtěný produkt určitý prostor pro společensky nebezpečné aktivity nového typu. Kriminalita páchaná v kybernetickém prostoru se vyznačuje řadou specifik. Kyberprostor nezná hranic. Kriminální útoky v kyberprostoru jsou velmi efektivní. Umožňují z jednoho místa zasáhnout chráněné zájmy na mnoha jiných místech ve velmi krátkém čase, v podstatě se zanedbatelnými finančními náklady a s minimálním nebezpečím okamžitého odhalení. Zatímco kybernetický prostor je prostorově neomezený, normy trestního práva platí jen na území příslušného státu. Z toho rozporu plyne výhoda pro pachatele využívající k páchání činů kyberprostor. Má-li se společnost efektivně bránit, je nezbytná určitá míra harmonizace trestněprávních norem (hmotných i procesních) a usnadnění mezinárodní spolupráce mezi státy v potírání společensky škodlivých jevů v kybernetickém prostoru. Jedním z nástrojů k dosažení nastíněného cíle je též Úmluva o počítačové kriminalitě (dále jen „Úmluva").

Úmluva je výsledkem čtyřleté práce expertů Rady Evropy, USA, Kanady, Japonska a dalších. Úmluva byla schválena Výborem ministrů Rady Evropy na jejím 109. zasedání 8. listopadu 2001. Otevřena k podpisu byla v Budapešti dne 23. listopadu 2001. V platnost vstoupila dne 1. července 2004. Ke dni 22. dubna 2008 Úmluvu podepsalo 44 států, z nichž jí však ratifikovalo jen 22 států, tedy pouhých 50 %. Z nečlenských států Rady Evropy Úmluvu podepsaly a zároveň ratifikovaly jen USA. Česká republika podepsala Úmluvu dne 9. února 2005. K její ratifikaci prozatím nedošlo, na rozdíl od našeho východního souseda, Slovenské republiky, která Úmluvu podepsala dne 4. února 2005, ratifikovala 8. ledna 2008 a pro níž vstoupí v platnost dnem 1. května 2008.

K Úmluvě byl přijat dodatkový protokol. Naskýtá se otázka, proč v relativně krátké době po přijetí Úmluvy byl přijat i dodatkový protokol. Vysvětlení je poměrně jednoduché. Návrhu Úmluvy jako celku se dostalo dosti výrazné podpory. Delegace některých států (zejména USA) však projevily určité pochybnosti nad návrhem kriminalizace některých činů s rasovým a xenofobním obsahem s poukazem na rozpor s právem na svobodu projevu. Jelikož nechtěli předkladatelé ohrozit přijetí Úmluvy, bylo upuštěno od snahy zařadit do 3. oddílu závazek kriminalizace rasové propagandy. Právě proto se oddíl Úmluvy s názvem „trestné činy související s obsahem" zabývá „jen" nezákonnou výrobou a šířením dětské pornografie. Právě proto byl závazek ke kriminalizaci některých činů s rasovým a xenofobním obsahem vyčleněn do samostatného protokolu. Dodatkový protokol byl otevřen k podpisu 28. ledna 2003, vstoupil v platnost 1. března 2005, podepsalo jej 32 států, z toho ratifikovalo jen 11. Česká republika, stejně jako např. Slovensko nebo USA, není signatářem dodatkového protokolu.

Cílem příspěvku je seznámit účastníky konference rámcově s obsahem Úmluvy, s využitím důvodové zprávy okomentovat některá její ustanovení a zhodnotit, nakolik je současná česká právní úprava v souladu s touto Úmluvou. Jelikož je v Poslanecké sněmovně Parlamentu České republiky projednáván návrh nového trestního zákoníku[1], připojuji u hmotněprávních ustanovení informaci, zda tento návrh na současné situace může něco změnit nebo nikoliv. V příspěvku se budu především věnovat ustanovením hmotněprávního charakteru, jejichž zavedení Úmluva vyžaduje. Ustanovení procesněprávního charakteru a závazky k mezinárodní spolupráci zmíním jen v obecné rovině.

2 Struktura Úmluvy

Úmluva, čítající 48 článků, se vedle preambule člení do 4 kapitol. Kapitola I. (používání pojmů) definuje pojmy „počítačový systém", „počítačová data", „poskytovatel služeb", „provozní data". Kapitola II. (opatření přijímaná na národní úrovni) upravuje závazky států v oblasti trestního práva hmotného (oddíl 1) i procesního (oddíl 2) včetně ustanovení o působnosti vnitrostátních norem (oddíl 3). Závazky na poli mezinárodní spolupráce jsou náplní kapitoly III. Závěrečná ustanovení nalezneme v kapitole IV.

3 Používání pojmů

Jelikož Úmluva používá v textu opakovaně některé pojmy, které se též pokouší definovat v čl. 1, bude vhodné hned na počátku podat jejich výčet a připojit pár poznámek.

[1] Srov. Sněmovní tisk č. 410, 5. volební období.

Počítačovým systémem se rozumí jakékoli zařízení nebo skupina vzájemně propojených nebo souvisejících zařízení, z nichž jedno nebo více provádí na základě programu automatické zpracování dat.

Počítačovými daty se rozumí jakékoli vyjádření skutečností, informací nebo pojmů ve formě vhodné pro zpracování v počítačovém systému, včetně programu vhodného k zajištění, aby počítačový systém vykonával určitou funkci.

Definice počítačových dat vychází z definice dat Mezinárodní organizace pro standardizaci (ISO). Výraz „vhodné ke zpracování“ zahrnuje data, která může počítačový systém zpracovat přímo.

Poskytovatelem služeb se podle Úmluvy rozumí:

a) jakýkoli veřejný nebo soukromý subjekt, který poskytuje uživatelům služby možnost komunikovat prostřednictvím počítačového systému,

b) jakýkoli jiný subjekt, který zpracovává nebo ukládá počítačová data pro tuto komunikační službu nebo uživatele této služby.

Jak je patrné z definice, je pojem poskytovatel služeb hodně široký. Zahrnuje soukromé i veřejné subjekty. Z hlediska Úmluvy je nerozhodné, zda uživatelé služeb tvoří uzavřenou skupinu (např. zaměstnanci jedné společnosti) či je služba poskytována veřejně. Stejně tak není rozhodující, zda je služba poskytována za úplatu anebo zdarma.

Podle bodu b) bude poskytovatelem služby i ten, kdo poskytuje vyrovnávací paměť pro internetové stránky (caching), zajišťuje provoz aplikace třetí osoby na vlastním technickém a programovém vybavení (hosting), tak služby, které poskytují připojení k síti. Definice však nezahrnuje pouhé poskytovatele obsahu, pokud takový obsah nenabízí také komunikaci nebo související služby zpracování dat.

Provozními daty se rozumí jakákoli počítačová data související s přenosem dat prostřednictvím počítačového systému, generovaná počítačovým systémem, který tvořil součást komunikačního řetězce, jež vyjadřují původ, cíl, trasu, dobu, objem, dobu trvání přenosu dat nebo druh použité služby.

Provozními daty jsou tedy data, která slouží k nasměrování komunikace od místa původu do místa určení (od odesílatele k adresátovi). Ve vztahu ke komunikaci, která tvoří tzv. obsahová data, plní pouze pomocnou úlohu. Jejich zachycení tedy nevede k odhalení obsahu sdělení. Provozní data jsou zapotřebí k vysledování zdroje komunikace jako výchozího bodu k získání dalších důkazů k prokázání trestného činu. Jejich specifikem je též skutečnost, že mohou trvat pouze krátkou dobu. Z toho pak plyne požadavek jejich co nejrychlejšího zjištění a zajištění.

Úmluva sama specifikuje, co tvoří provozní data. Jsou to data o původu komunikace (tj. telefonní číslo, IP adresa nebo jiná identifikace komunikačního zařízení, jemuž poskytovatel služby poskytuje službu), určení komunikace (tedy

telefonní číslo, IP adresa nebo jiná identifikace komunikačního zařízení, jemuž se příslušné komunikace, sdělení, zasílají), trasa, čas, délka, trvání a typ příslušné služby. Typem příslušné služby se rozumí typ (druh) služby používaný v síti, např. přenos souborů, elektronická pošta nebo okamžité posílání a příjem zpráv.

4 Závazky hmotněprávního charakteru

Úmluva obsahuje znaky 9 trestných činů, které dělí do 4 kategorií. Mimo to se Úmluva týká některých otázek základů trestní odpovědnosti. Vyžaduje se trestnost účastenství, pokusu trestného činu a zavedení alespoň tzv. nepravé odpovědnost právnických osob za trestné činy upravené v Úmluvě. Sankce za takové činy mají být účinné, přiměřené a odstrašující. V otázkách základů trestní odpovědnosti je naše úprava s výjimkou odpovědnosti právnických osob v souladu s požadavky Úmluvy. Problém nemáme ani se stanovením místní působnosti trestního zákona na takové činy.

Po obecném výkladu, jež je společný všem činům, u nichž se vyžaduje kriminalizace, se zaměříme pouze na některé z nich.

subsectionObecný výklad

Při vymezení znaků jednotlivých činů, jejichž kriminalizaci Úmluva vyžaduje, je téměř vždy uveden výslovně znak **protiprávnosti** slovy „neoprávněně“, „protiprávně“.[2] Tím má být zdůrazněno, že mohou existovat i činnosti dovolené, byť vykazují všechny další znaky uvedených činů. Typickými případy, kdy je vyloučena protiprávnost jsou např. činnosti právním řadem přikázané nebo dovolené, jednání v nutné obraně a v krajní nouzi, souhlas poškozeného, přípustné riziko apod.

V českém trestním právu je protiprávnost znakem každé skutkové podstaty každého trestného činu, byť bývá výslovně uvedena jen u malé skupiny z nich. Je-li uvedena, slovní vyjádření se v jednotlivých skutkových podstatách liší, děje se tak např. slovy „neoprávněně“ (§ 118 TZ), „bez povolení“ (§ 185 TZ), „v rozporu s právním předpisem“ (§ 148a TZ). Není-li protiprávnost výslovně vyjádřena, neznamená to, že se k trestnosti činu nevyžaduje. Vždyť např. i trestný čin vraždy výslovně protiprávnost nevyjadřuje ve znění skutkové podstaty („kdo jiného úmyslně usmrtí“). Přitom bezesporu existují případy, kdy úmyslné usmrcení jiného nebude trestným činem v důsledku absence protiprávnosti (např. usmrcení jiného v mezích nutné obrany). Lze tedy uzavřít, že protiprávnost je znakem trestného činu vždy, i když není výslovně vyjádřena ve znění skutkové podstaty trestného činu.

Ani jeden z definovaných činů **nelze spáchat z nedbalosti**. Úmluva vyžaduje vždy úmyslnou formu zavinění. U některých z činů jde požadavek úmyslné

[2] Pouze v čl. 10 je protiprávnost vyjádřena jako „porušení autorského práva“.

formy zavinění ještě dál, požaduje se nejen úmysl ve vztahu k jednání, ale též aby úmysl zahrnoval i dosažení určitého cíle sledovaného popsaným jednáním. Pokud by absentoval úmysl ve vztahu k dosažení takového cíle, bylo by jednání beztrestné, byť bylo provedeno úmyslně. Teorie mluví o tzv. **druhém (specifickém) úmyslu** či úmyslu přesahujícím objektivní stránku skutkové podstaty trestného činu. Typickým příkladem specifického úmyslu je úmysl získat majetkový prospěch (čl. 8). Uveďme si jiný příklad, a to ustanovení čl. 6 (zneužití zařízení). Trestným činem má být např. zpřístupnění jinému počítačového hesla, pomocí něhož lze získat přístup k počítačovému systému, ale pouze za podmínky, že tak pachatel učinil v úmyslu, aby jej bylo využito pro účely spáchání některého z činů uvedených v čl. 2 až 5. Rozebereme-li si uvedený čin z hlediska zavinění, pak se vyžaduje, aby pachatel:

1. úmyslně jinému zpřístupnil počítačové heslo, pomocí něhož lze získat přístup k počítačovému systému (úmysl),

2. tak učinil v úmyslu, aby heslo bylo využito pro účely spáchání některého z činů uvedených v čl. 2 až 5 (specifický úmysl).

Ke kriminalizaci proto nestačí, jestliže osoba úmyslně zpřístupní jinému počítačové heslo, jestliže tak nečiní ve specifickém úmyslu, aby ho bylo využito pro účely spáchání některého z činů uvedených v čl. 2 až 5.

Úmluva v řadě případů umožňuje trestnost omezit jen na závažnější případy, tj. že smluvní stát bude k trestnosti vyžadovat naplnění tzv. **dodatečné náležitosti** (**další kvalifikační okolnosti**) – srov. např. čl. 2 věta druhá; čl. 3 věta druhá, čl. 6 odst. 1 písm. b), čl. 7, čl. 9 odst. 3 Úmluvy.

Dodatečnými náležitostmi jsou např. tyto:

- porušení bezpečnostních opatření,
- nečestný úmysl pachatele,
- čin je spáchán ve vztahu k počítačovému systému, který je propojen s jiným počítačovým systémem,
- jednání může vést ke vzniku závažnější škody,
- držení je trestné jen při držení určitého počtu věcí.

Jestliže hodlá smluvní stát využít omezení trestnosti, musí v souladu čl. 40 učinit písemné prohlášení, že si ponechává možnost požadovat dodatečné náležitosti. Např. Slovenská republika využila možnosti stanovit dodatečné náležitosti v případě čl. 2 (neoprávněný přístup).

Další omezení v rozsahu plnění závazků vyplývajících z Úmluvy znamená **možnost** smluvních států **učinit výhradu k aplikaci** některých taxativně vyjmenovaných článků (viz čl. 42).

Objektem činů definovaných v Úmluvě je ochrana důvěrnosti, neporušenosti a použitelnosti počítačových systémů nebo dat (čl. 2 až 6), ochrana před sexuálním zneužíváním dětí (čl. 9), ochrana literární, umělecké a vědecké tvůrčí činnosti a jejich výsledků (čl. 10). Pokud jde o trestné činy související s počítači (čl. 7 a 8), individuální objektem je ochrana počítačových dat, údajů a systémů, pokud jde o jejich důvěrnost. Na rozdíl od čl. 2 až 6 je změna dat, údajů a systémů prostředkem ke spáchání jiných trestných činů. Vždy zde tedy bude přítomen zároveň další chráněný zájem (např. ochrana majetku – čl. 8).

Úmluva stanoví znaky těchto trestných činů:

1. *Trestné činy proti důvěrnosti, integritě a dostupnosti počítačových dat a systémů*
   a) **Neoprávněný přístup (čl. 2)**
   b) **Neoprávněné zachycení informací (čl. 3)**
   c) **Zásah do dat (čl. 4)**
   d) **Zásah do systému (čl. 5)**
   e) **Zneužití zařízení (čl. 6)**
2. *Trestné činy související s počítači*
   a) **Falšování údajů související s počítači (čl. 7)**
   b) **Podvod související s počítači (čl. 8)**
3. *Trestné činy související s obsahem*
   a) **Trestné činy související s dětskou pornografií (čl. 9)**
4. *Trestné činy související s porušením autorského práva a práv příbuzných autorskému právu*
   a) **Trestné činy související s porušením autorského práva a práv příbuzných autorskému právu (čl. 10)**

4.1 Vybrané jednotlivé trestné činy

4.1.1 Neoprávněný přístup (čl. 2)

Objektem tohoto činu je ochrana před ohrožením bezpečnosti počítačových systémů a dat. Potřeba ochrany odráží zájem korporací a jednotlivců řídit, provozovat a kontrolovat své systémy nenarušovaným způsobem a bez zábran. Pouhé

neoprávněné vniknutí, tj. „průnikaření“ (*„hacking“; „cracking“; „computer trespass“*) by již v zásadě mělo být podle čl. 2 samo o sobě protiprávní.

Objektivní stránka spočívá v přístupu do počítačového systému nebo jeho části. Trestný má být již samotný průnik (neoprávněné vniknutí, vstup) do počítačového systému. Průnik do tohoto systému je často jen přípravou k závažnějšímu činu.

Pachatelem může být kdokoliv. **Zavinění** se vyžaduje úmyslné.

Smluvní státy mohou prostřednictvím následujících **dodatečných kvalifikačních okolností** omezit trestnost neoprávněného přístupu:

a) porušení bezpečnostních opatření,

b) úmysl získat počítačová data nebo jiný nečestný úmysl, nebo

c) čin je proveden ve vztahu k počítačovému systému, který je propojen s jiným počítačovým systémem.

Podle českého trestního zákona (§ 257a TZ – poškození a zneužití záznamu na nosiči informací) je trestné získání přístupu k nosiči informací, pokud jsou kumulativně splněny dvě podmínky:

1. pachatel tak činí v úmyslu způsobit jinému škodu nebo jinou újmu nebo získat sobě nebo jinému neoprávněný prospěch, což by ještě bylo v souladu s Úmluvou pokryto možností aplikovat dodatečné okolnosti,
2. pachatel se dopustí některého z alternativně uvedených jednání: a) takových informací neoprávněně užije; b) informace zničí, poškodí, změní nebo učiní neupotřebitelnými; nebo c) učiní zásah do technického nebo programového vybavení počítače nebo jiného telekomunikačního zařízení. Tyto zužující podmínky však již nejsou v souladu s Úmluvou.

Návrh trestního zákoníku upravuje v § 228 trestný čin **Neoprávněný přístup k počítačovému systému a nosiči informací**. V odstavci 1 je trestné, jestliže někdo překoná bezpečnostní opatření, a tím neoprávněně získá přístup k počítačovému systému nebo k jeho části. Tím bude trestní zákoník uveden do souladu s Úmluvou, byť si Česká republika zachová omezení trestnosti na případy, kdy dochází k neoprávněnému přístupu překonáním bezpečnostních opatření. Proto bude nutné učinit prohlášení podle čl. 40 Úmluvy.

4.1.2 Neoprávněné zachycení informací (čl. 3)

Objektem čl. 3 je ochrana práva na soukromí datové komunikace. Ve své podstatě lze přirovnat tento čin k tradičním odposlechům. Článek 3 se vztahuje na všechny formy přenosu elektronických dat (např. telefonem, faxem, elektronickou poštou, přenosem souborů). Není však zamýšlena kriminalizace používání běžných obchodních praktik jako jsou „cookies“ jako takových.

Objektivní stránka spočívá v zachycení neveřejných přenosů počítačových dat pomocí technických prostředků, a to z nebo v rámci počítačového systému, včetně zachycení elektromagnetických emisí z počítačového systému, obsahující taková počítačová data. Znak „neveřejnosti“ se zde vztahuje na způsob přenosu, nikoliv na obsahovou stránku přenosu. I neveřejná data mohou být přenášena veřejně. V takovém případě se na ně tento článek nevztahuje. A naopak, veřejně dostupná informace může být mezi účastníky přenášena důvěrně (neveřejně). Např. komunikace zaměstnanců, ať již slouží pracovním účelům či ne, která má povahu „neveřejných přenosů počítačových dat“, se rovněž chrání před neoprávněným odposlechem podle článku 3. To však neznamená, že by vnitrostátní právo nemohlo umožnit odposlech takové komunikace. Komunikace v podobě přenosu počítačových dat může probíhat např. uvnitř jednoho počítačového systému (například tok dat z CPU k obrazovce nebo tiskárně), mezi dvěma počítačovými systémy patřícími téže osobě, anebo mezi počítačem a osobou (například prostřednictvím klávesnice). Elektromagnetické vyzařování může počítač emitovat během svých operací. Takové vyzařování se nepovažuje za „data“ ve smyslu definice uvedené v článku 1. Lze však z něj data rekonstruovat. Proto je podle tohoto ustanovení odposlech dat z elektromagnetického vyzařování pocházejícího z počítačového systému trestným činem.

Pachatelem může být kdokoliv, **zavinění** se vyžaduje úmyslné.

Smluvní státy mohou prostřednictvím následujících **dodatečných kvalifikačních okolností** omezit trestnost na případy nečestného úmyslu nebo je-li čin proveden ve vztahu k počítačovému systému, který je propojen s jiným počítačovým systémem.

Český trestní zákon upravuje v § 239 a 240 trestný čin porušování tajemství dopravovaných zpráv. Je vhodné připomenout, že v době přijetí trestního zákona (v r. 1961), zcela jistě zákonodárce nezamýšlel aplikaci § 239, 240 na případy přenosu počítačových dat. Ani pozdější novelizace nesledovaly takový účel. Podle § 239 odst. 1 písm. b) se dopustí tohoto trestného činu, kdo úmyslně poruší tajemství zprávy podávané telefonem, telegrafem nebo jiným takovým veřejným zařízením. Použití tohoto ustanovení na zachycení neveřejných přenosů počítačových dat pomocí technických prostředků, je problematické. Přenos počítačových dat nemusí být patrně „zprávou“ ve smyslu § 239 TZ. Navíc zůstává nekriminalizován odposlech elektromagnetických emisí z počítačového systému obsahujícího taková počítačová data.

Návrh trestního zákoníku je důslednější a v § 180 pamatuje i na přenos počítačových dat včetně možnosti odposlechu elektromagnetického vyzařování. Podle § 180 odst. 1 písm. c) Návrhu se trestného činu porušení tajemství dopravovaných zpráv dopustí, kdo úmyslně poruší tajemství neveřejného přenosu počítačových dat do počítačového systému, z něj nebo v jeho rámci, včetně elektromagnetického vyzařování z počítačového systému, přenášejícího taková počítačová data. Soulad s Úmluvou bude v případě přijetí Návrhu zajištěn. Při-

způsobení české právní úpravy Úmluvě bude úplné, dokonce nebude třeba využít možnosti dodatečných okolností.

4.1.3 Zásah do dat (čl. 4)

Objektem čl. 4 je ochrana počítačových dat a počítačových programů před úmyslným způsobením škody – porušením integrity, nesprávným fungování nebo neoprávněným využíváním uložených počítačových dat nebo počítačových programů.

Objektivní stránka spočívá v poškození, vymazání, zhoršení kvality, změně nebo potlačení počítačových dat. Potlačením dat se rozumí zabránění jejich upotřebitelnosti. Pod toto ustanovení lze subsumovat činnost „zákeřných programů" jako jsou viry a trojské koně. Za **oprávněnou** (nikoliv protiprávní) **činnost** bude třeba podle důvodové zprávy považovat testování ochrany bezpečnosti počítačového systému schválené vlastníkem nebo operátorem, anebo rekonfigurace operačního systému počítače, která probíhá v případech, kdy operátor systému získá nové programové vybavení (například software umožňující přístup k Internetu a vyřazující z činnosti podobné, dříve instalované programy). Modifikace provozních dat za účelem usnadnění anonymních komunikací (například činnost systémů pro anonymní (pře)posílání emailové pošty – „*anonymous remailer systems*") anebo modifikace dat za účelem zabezpečení komunikací (například zašifrování) by se v zásadě měly považovat za legitimní ochranu soukromí, a tedy za úkony provedené oprávněně. Smluvní strany však mohou kriminalizovat určitá zneužití týkající se anonymních komunikací, jako například změnu informace v záhlavích paketů („*packet header information*"), aby se skryla totožnost pachatele trestného činu.

Pachatelem může být kdokoliv. **Zavinění** se vyžaduje úmyslné.

Smluvní státy mohou **formou výhrady** omezit trestnost pouze na činy, které vedou ke vzniku závažné újmy. Možnost **dodatečných okolností** není připuštěna.

Kdo získá přístup k nosiči informací a v úmyslu způsobit jinému škodu nebo jinou újmu nebo získat sobě nebo jinému neoprávněný prospěch informace zničí, poškodí, změní nebo učiní neupotřebitelnými se dopustí trestného činu **poškození a zneužití záznamu na nosiči informací podle § 257a odst. 1 písm. b) TZ**. V porovnání s požadavky vyplývajícími z Úmluvy lze poznamenat tři odlišnosti. 1) Úmluva mimo jiné požaduje kriminalizaci snížení kvality dat a potlačení dat. Nejsem si zcela jist, zda změna dat je natolik široký pojem, aby pokryla všechny případy snížení kvality dat. Naopak, potlačení dat lze patrně považovat za učinění dat neupotřebitelnými. 2) Úmluva připouští omezení kriminalizace na případy, kdy dojde k vážné újmě, český trestní zákon omezuje trestnost na případy existence úmyslu způsobit jinému škodu nebo jinou újmu nebo získat sobě nebo jinému neoprávněný prospěch.

3) Předpokladem aplikace § 257a odst. 1 písm. b) TZ je získání přístupu k nosiči informací. Tato podmínka je zjevně nadbytečná z pohledu Úmluvy. Nečiní však problém, jelikož získáním přístupu se rozumí takové jednání, které umožní pachateli volnou dispozici s nosičem informací a využití jeho informačního obsahu, nehledě na to, zda jde o přístup oprávněný nebo neoprávněný.

Ustanovení § **228 odst. 2 písm. b) Návrhu** je přizpůsobeno znění čl. 4 Úmluvy, když za trestný čin prohlašuje jednání, kterým někdo získá přístup k počítačovému systému nebo k nosiči informací a data uložená v počítačovém systému nebo na nosiči informací neoprávněně vymaže nebo jinak zničí, poškodí, změní, potlačí, sníží jejich kvalitu nebo je učiní neupotřebitelnými. Navrhovatel se patrně domnívá, že potlačení dat je něco jiného než činění dat neupotřebitelnými.

4.1.4 Zásah do systému (čl. 5)

Objektem čl. 5 je ochrana zájmu operátorů a uživatelů počítačových nebo telekomunikačních systémů na řádném fungování těchto systémů. Čin lze označit též za tzv. počítačovou sabotáž.

Objektivní stránka spočívá v závažném narušení fungování počítačového systému vložením, přenesením, poškozením, vymazáním, zhoršením kvality, změnou nebo potlačením počítačových dat.

Přílišné kriminalizaci má zabránit požadavek, aby narušení fungování systému bylo závažné. Sem bude spadat též odesílání dat určitému systému v takové podobě, objemu nebo frekvenci, že to má za následek významný škodlivý vliv na schopnost vlastníka nebo operátora využívat systém nebo komunikovat s ostatními systémy, například prostřednictvím programů, které generují útoky typu odepření služby („*denial of service*“), škodlivé („*malicious*“) kódy jako jsou viry, které zabraňují chodu systému, nebo jej podstatně zpomalují, anebo programy, které odesílají velká množství elektronické pošty příjemci, aby zablokovaly komunikační funkce systému. Odesílání nevyžádané elektronické pošty ve velkém množství nebo s vysokou frekvencí známé pod označením „spamming“ lze tudíž podle čl. 5 Úmluvy kriminalizovat pouze tehdy, je-li závažně omezena funkčnost komunikace. Zde je nutné připomenout, že Úmluva tvoří pouze minimální konsensus, pokud jde o povinnost státu stíhat vyjmenované činy. Nic v Úmluvě nebrání státům, aby stíhaly širší okruh jednání. Je na jednotlivých smluvních státech, aby zvážili nutnost kriminalizace spammingu s ohledem na subsidiární úlohu trestněprávní ochrany.

Pachatelem může být kdokoliv. **Zavinění** se vyžaduje úmyslné.

Dodatečné kvalifikačních okolnosti nejsou stanoveny. Textace odst. 1 totiž umožňuje smluvním státům dostatečně omezit trestnost jednání, když vyžaduje, aby omezení funkčnosti bylo „vážné“, čímž ponechává na státech, aby

rozhodly o míře, v níž by mělo být fungování systému omezeno (např. zčásti nebo úplně, dočasně nebo trvale).

Mnohokrát zmiňovaný § **257a TZ** umožňuje kriminalizovat zásah do technického nebo programového vybavení počítače (nebo jiného telekomunikačního zařízení), přičemž počítačem se rozumí každá funkční jednotka, která může provádět výpočty všech číselných aritmetických a logických operací bez lidského zásahu a podle počítačového programu. Je to zařízení na zpracování, uchovávání a využití informací rozmanitého rázu, které převádí na číselné kódy. Počítačovým systémem se podle čl. 1 písm. a) Úmluvy rozumí jakékoli zařízení nebo skupina vzájemně propojených nebo souvisejících zařízení, z nichž jedno nebo více provádí na základě programu automatické zpracování dat. Porovnáním obsahu obou pojmů „počítač“ a „počítačový systém“ zjistíme, že jsou v mnoha směrech shodné, pojem počítačový systém je bezpochyby širší (srov. výše III. kapitolu). Úmluva však požaduje kriminalizaci narušení fungování počítačového systému, trestní zákon se spokojuje s učiněním zásahu do technického nebo programového vybavení počítače, aniž by bylo vyžadováno narušení jeho fungování. Trestní zákon navíc požaduje, aby tak pachatel učinil v úmyslu způsobit jinému škodu nebo jinou újmu nebo získat sobě nebo jinému neoprávněný prospěch. Shledávám, že současný trestní zákon nechrání specificky narušení počítačového systému, jak vyžaduje Úmluva.

Návrh trestního zákoníku stanoví jako okolnost podmiňující použití vyšší trestní sazby podle § 228 odst. 3 písm. b), jestliže pachatel spáchal trestný čin podle odst. 1 nebo 2 v úmyslu neoprávněně omezit funkčnost počítačového systému nebo jiného technického zařízení pro zpracování dat. Jelikož odst. 1 a 2 obsahují nejen znaky uvedené v čl. 5, jde navrhovaná úprava nad rámec závazku vyplývajícího z Úmluvy.

4.1.5 Zneužití zařízení (čl. 6)

Objektem je zájem na ochraně společnosti a osob před možným ohrožením vyplývajícím z nekontrolované výroby, prodeje, jiné distribuci, držení věcí, jež primárně slouží k trestné činnosti v kybernetickém prostoru (čl. 2 až 5) nebo pomocí nichž lze získat přístup k počítačovému systému nebo jeho části. V tomto ustanovení dochází ke kriminalizaci jednání, která jsou ve své materiální povaze přípravným jednáním k trestné činnosti v kyberprostoru, tj. typicky předcházejí této trestné činnosti. Ke spáchání některého z trestných činů vymezených v čl. 2 až 5 je často zapotřebí opatření přístupových prostředků jako jsou různé hackerské nástroje.

Objektivní stránka spočívá v:

a) výrobě, prodeji, obstarání k užívání, dovozu, distribuci nebo jiném zpřístupnění:

(i) zařízení, včetně počítačového programu, určeného nebo přizpůsobeného **primárně** pro účely spáchání kteréhokoli trestného činu stanoveného v souladu s články 2 až 5;

(ii) počítačového hesla, přístupového kódu nebo podobného údaje, pomocí něhož lze získat přístup k počítačovému systému nebo jeho části,

b) držbě jakékoli věci uvedené v písmenu a) bodech (i) a (ii).

Zpřístupněním na rozdíl od rozšiřování nevyžaduje aktivní přístup pachatele. Půjde např. o umístění hackerského programu na internet, aby si jej mohli stáhnout jiní uživatelé internetu. Postačí též uvedení tzv. hyperlinku pro usnadnění přístupu k takovému nástroji. Při stanovení znaků tohoto trestného činu se dlouze debatovalo, zdali nemá být trestnost omezena pouze na zařízení, která jsou výlučně konstruována k páchání trestné činnosti, aby se zabránilo nadbytečné kriminalizaci a obtížím v dokazování, pokud jde o prostředky, které mohou mít nejen ilegální, ale též legální použití. Konsensus byl nalezen, pokud jde o zařízení, která jsou objektivně vytvořena nebo přizpůsobena primárně za účelem spáchání trestného činu. To samo o sobě obvykle vyloučí zařízení s dvojím využitím (legálním i nelegálním).

Úmluva raději výslovně a jasně stanoví, že se povinnost kriminalizovat nevztahuje na výrobu, prodej, obstarání k užívání, dovoz, distribuci nebo jiné zpřístupnění nebo držbě uvedené v odstavci 1, pokud nesleduje účel spáchání trestného činu stanoveného v souladu s články 2 až 5 Úmluvy, jako je tomu například, v případě oprávněného testování nebo ochrany počítačového systému. Tento požadavek je zřejmě nadbytečný, neboť je obsažen již ve znaku „neoprávněně".

Pachatelem může být kdokoliv. **Zavinění** se vyžaduje úmyslné. U obou alternativ pod písmeny a) i b) se vyžaduje ještě specifický úmysl, kterým je využití zařízení pro účely spáchání kteréhokoli z trestných činů stanovených v článcích 2 až 5.

Dodatečnou kvalifikační okolností v případě držení podle písm. b) je možnost smluvního státu určit minimální počet takových věcí pro vznik trestní odpovědnosti.

Každá smluvní strana si může navíc **vyhradit právo** neuplatňovat odstavec 1 čl. 6, pokud se taková výhrada netýká prodeje, distribuce nebo jiného zpřístupňování věcí uvedených v odstavci 1 písm. a) bodu (ii) čl. 6.

Český trestní zákon nemá skutkovou podstatu, která by kriminalizovala jednání vymezené v čl. 6 Úmluvy. Ani obstarání zařízení za účelem spáchání trestného činu poškození a zneužití záznamu na nosiči informací (§ 257a) není trestné, neboť takové přípravné jednání nevykazuje všechny znaky přípravy k trestnému činu (§ 7 odst. 1 TZ). Příprava je totiž trestná jen u zvlášť závažných trestných činů (§ 41 odst. 2 TZ), kterým trestný čin poškození a zneužití záznamu na nosiči informací podle § 257a není.

Návrh trestního zákoníku na tuto mezeru reaguje novým § 229 (Opatření a přechovávání přístupového zařízení a hesla k počítačovému systému a jiných takových dat). Znaky tohoto trestného činu jsou vymezeny natolik široce, že kryjí všechny podoby jednání podle čl. 6 Úmluvy. Návrh na rozdíl od Úmluvy nestanoví, že zařízení je určeno nebo přizpůsobeno primárně pro účely spáchání uvedených trestných činů. Zdá se, že navrhovaná úprava je užší, neboť vyžaduje, aby zařízení bylo vytvořeno nebo přizpůsobeno k neoprávněnému přístupu. Nebude patrně pokrývat případy, kdy zařízení může sloužit, byť jen zčásti, účelům legálním.

4.1.6 Trestné činy související s dětskou pornografií (čl. 9)

Objektem je ochrana před sexuálním zneužíváním dětí. V odst. 2 písm. b) a c) čl. 9, ačkoliv se netýká sexuálního zneužívání skutečného dítěte, se poskytuje ochrana před chováním, které by mohlo vést k podněcování nebo svádění dětí ke skutečnému sexuálnímu zneužívání.

Objektivní stránka spočívá v:

a) výrobě dětské pornografie pro účely její distribuce prostřednictvím počítačového systému;

b) nabízení nebo zpřístupňování dětské pornografie prostřednictvím počítačového systému;

c) distribuci nebo přenášení dětské pornografie prostřednictvím počítačového systému;

d) obstarávání dětské pornografie prostřednictvím počítačového systému pro sebe nebo pro jinou osobu;

e) držbě dětské pornografie v počítačovém systému nebo na médiu pro ukládání počítačových dat.

Zpřístupnění zahrnuje též tvorbu nebo kompilaci odkazů („*hyperlinks*“) na stránky s dětskou pornografií, aby se tak usnadnil přístup k dětské pornografii. Rozdíl mezi *distribucí a přenášením* dětské pornografie spočívá ve způsobu šíření. Jde-li pouze o zaslání dětské pornografie jiné osobě, pak se jedná o přenášení. Při distribuci jde o aktivní šíření materiálu. Obsah pojmu pornografický materiál je ponechán na vnitrostátní úpravě. Nemusí zahrnovat materiál s uměleckou, lékařskou, vědeckou nebo podobnou hodnotou.

Dítětem se rozumí dítě mladší 18ti let. Pojem „*dětská pornografie*“ zahrnuje pornografické materiály, které vizuálně zobrazují a) dítě provádějící viditelný sexuální akt; b) osobu, jež vyhlíží jako dítě, provádějící viditelný sexuální akt; c) realistické zobrazení dítěte provádějícího viditelný sexuální akt. Pojem *viditelný sexuální akt* pokrývá přinejmenším tyto skutečné nebo simulované situace:

a) pohlavní styk, včetně styku genitálně-genitálního, orálně-genitálního, análně-genitálního nebo orálně-análního, mezi dětmi nebo mezi dospělým a dítětem, téhož nebo opačného pohlaví; b) sodomie; c) masturbace; d) sadistické nebo masochistické praktiky v sexuálním kontextu; nebo e) necudné (lascivní) předvádění genitálu nebo pubické krajiny dítěte. Není rozhodné, zda je zobrazené chování skutečné nebo předstírané (simulované).

Pachatelem může být kdokoliv. **Zavinění** se vyžaduje úmyslné.

Dodatečnou kvalifikační okolností může být snížení věkové hranice dítěte až na 16 let.

Smluvní státy jsou oprávněny **učinit výhradu**, která se může týkat neuplatnění (zcela nebo zčásti) trestnosti obstarávání dětské pornografie prostřednictvím počítačového systému pro sebe nebo pro jinou osobu nebo držby dětské pornografie v počítačovém systému nebo na médiu pro ukládání počítačových dat. Dále lze učinit výhradu, že za dětskou pornografii nebude ve vnitrostátním právu považovány pornografické materiály, které vizuálně zobrazují osobu, jež vyhlíží jako dítě provádějící viditelný sexuální akt nebo které realisticky zobrazují dítě provádějícího viditelný sexuální akt.

Novelizace **českého trestního zákona** v roce 2007 (zákon č. 271/2007 Sb., účinný od 1. prosince 2007) změnila § 205 TZ, do té doby nesoucí název „Ohrožování mravnosti“. Změna se týkala nejen názvu tohoto trestného činu (nyní „Šíření pornografie“), ale též jeho znaků. Zároveň došlo novelou k zavedení nových trestných činů. V § 205a je kriminalizováno přechovávání dětské pornografie a v § 205b zneužití dítěte k výrobě pornografie. Trestné je podle § 205 odst. 2 písm. a) TZ, jestliže někdo vyrobí, doveze, vyveze, proveze, nabídne, činí veřejně přístupným, zprostředkuje, uvede do oběhu, prodá nebo jinak jinému opatří fotografické, filmové, počítačové, elektronické nebo jiné pornografické dílo, které zobrazuje nebo jinak využívá dítě. Kdo přechovává fotografické, filmové, počítačové, elektronické nebo jiné pornografické dílo, které zobrazuje nebo jinak využívá dítě, dopustí se trestného činu přechovávání dětské pornografie podle § 205a TZ. Oproti původnímu poslaneckému návrhu (sněmovní tisk č. 114, 5. volební období) není kriminalizováno šíření pornografického díla, které zobrazuje osobu, jež se jeví být dítětem. Proto v případě ratifikace Úmluvy bude nezbytné v tomto směru vznést výhradu (viz shora).

Při přípravě novely bylo využito **Návrhu trestního zákoníku**, proto současná právní úprava se v zásadě shoduje s návrhem.

5 Závazky procesněprávního charakteru

Vytvoření nových či modifikace stávajících znaků trestných činů, které postihují nebezpečné jevy v kyberprostoru je pouze prvním krokem k jejich efektivnímu postihu. Nové způsoby zneužívání kybernetického prostoru vyžadují i vytvoření

nových, popř. úpravu stávajících (tradičních) procesních institutů, které umožní odhalit pachatele a zajistit důkazy, jež mohou vést k jeho usvědčení. Je totiž nutné si uvědomit, že v prostředí, které vznikne propojením mnoha sítí, je nesnadné identifikovat pachatele, zjistit rozsah a následky trestného činu. K tomu přistupuje fakt, že elektronická data jsou charakteristická svoji nestálostí – mohou být v okamžiku změněna nebo i zničena. Z této charakteristiky vyplývá i potřeba, aby odhalení pachatele a zajištění důkazů probíhalo velmi rychle a v utajení. V souvislosti s potřebou zajistit elektronická data se při přípravě Úmluvy široce diskutovalo, zda by neměla být uložena poskytovatelům služeb povinnost po určitou dobu shromažďovat a uchovávat data. Nakonec žádné obdobné ustanovení v Úmluvě nenajdeme, neboť konsensu nebylo dosaženo.

Elektronická data se dají rozdělit do několika skupin. Podle toho, jaké informace obsahují, je lze rozdělit zpravidla na **data provozní**, **data obsahová** a **data o odběratelích**. Úmluva definuje *provozní data* (jakákoli počítačová data související s přenosem dat prostřednictvím počítačového systému, generovaná počítačovým systémem, který tvořil součást komunikačního řetězce, jež vyjadřují původ, cíl, trasu, dobu, objem, dobu trvání přenosu dat nebo druh použité služby). Nezajímá nás tedy obsah komunikace, ale jak komunikace probíhá. *Obsahová data* definována nejsou. Lze je jistě vymezit negativně, jako data, která nejsou provozního charakteru. To samo o sobě k jejich definici nestačí, neboť jiná než provozní data zahrnují též data o odběratelích. Z označení „obsahová data“ plyne, že nás zajímá, co je sdělováno (význam a účel sdělení, zprávy či informace předávané komunikací), tedy obsah komunikace. *Informace o odběrateli* zahrnují různé druhy informací o využívání služby a o uživateli této služby, pokud se nejedná o data provozní nebo obsahová. Jejich prostřednictvím lze zjistit: typ využívané komunikační služby, technické prostředky používané pro tuto službu a dobu trvání služby; totožnost uživatele, jeho poštovní nebo geografickou adresu, telefonní a jiné přístupové číslo a informace o fakturaci a platbách; jakékoli jiné informace o místu instalace komunikačního zařízení (srov. čl. 18 odst. 3). Rozlišení dat provozních, obsahových a o odběratelích má význam pro určení míry zásahu do soukromí osob, čemuž by měly odpovídat záruky proti případnému zneužití.

Dalším dělením počítačových dat podle okamžiku jejich výskytu může být klasifikace na **data uložená**, **data v procesu komunikace**, a **data, jež mají být teprve komunikována**.

Úmluva zavazuje státy k přijetí opatření, jež umožní efektivní využití počítačových dat k odhalení a usvědčení pachatele. Za tím účelem stanoví následující **procesní opatření**:

1. bezodkladné uchování uložených počítačových dat (čl. 16),
2. bezodkladné uchování a částečné poskytnutí (zpřístupnění) provozních dat (čl. 17),

3. příkaz k vydání (čl. 18),

4. prohlídka a zajištění uložených počítačových dat (čl. 19),

5. shromažďování provozních dat v reálném čase (čl. 20),

6. zachycení dat o obsahu (čl. 21).

Než přejdeme k charakteristice jednotlivých procesních opatření, je třeba přičinit několik poznámek.

Nejprve je třeba vyjasnit, **u kterých trestných činů lze použít, resp. musí být umožněno použití výše uvedených procesních opatření**. Úmluva zavazuje státy, aby umožnily jejich použití na:

a) trestné činy, k jejichž kriminalizaci zavazuje Úmluva (viz čl. 2 až 11),

b) jiné trestné činy, které byly spáchány pomocí počítačového systému,

c) shromažďování důkazů v elektronické podobě o trestném činu (zde bez ohledu na to, o jaký trestný čin se jedná).

Z uvedeného rozsahu existují **dvě výjimky** (srov. čl. 20 a 21).

Procesní opatření tak, jak jsou Úmluvou vymezena, významně zasahují do práva a svobod osob. Proto je nutné, aby takový zásah byl odůvodněn, podléhal kontrole, byl přiměřený a co nejméně zasahoval do práv třetích osob (zásada subsidiarity a přiměřenosti).

Procesní opatření podle čl. 16 až 21 jsou v pravomoci „**příslušného orgánu**“, kterým se rozumí soudní, správní nebo jiný orgán vynucování práva, který je vnitrostátním právem zmocněn nařizovat, schvalovat nebo provádět procesní opatření za účelem shromažďování nebo předkládání důkazů. Je též třeba zdůraznit, že opatření je možné použít vždy jen **v konkrétním trestním řízení**. Nelze proto např. nařídit „odposlech“ obsahových dat bez souvislosti s konkrétním trestním řízením.

Stanovení některých opatření, např. příkazu k vydání podle čl. 18, může mít i **význam pro třetí strany** (např. správce dat), kteří jsou jinak ochotni dobrovolně poskytnout součinnost orgánům činným v trestním řízení, neboť je zbavuje jakékoliv smluvní nebo mimosmluvní odpovědnosti ve vztahu k osobám, jichž se data týkají.

Česká právní úprava dostatečně neodráží veškeré závazky, které vyplývají z Úmluvy. Je to patrně dáno tím, že problematice není věnována větší pozornost ze strany trestněprávních odborníků. Některé aspekty jsou upraveny v zákoně č. 127/2005 Sb., o elektronických komunikacích (dále jen „ZoEK“). Trestní řád však nemá ustanovení týkající se specificky počítačových dat. Využívají se proto

stávající instituty trestního řádu jako např. povinnost k vydání věci (§ 78), odnětí věci (§ 79), zadržení, otevření a sledování zásilky (§ 86–87c), odposlech a záznam telekomunikačního provozu (§ 88, 88a), které však s ohledem na specifika počítačových dat nejsou vždy vyhovující. O poznání lépe je na tom náš východní soused – Slovenská republika. Slovenský trestní řád z roku 2005 (zákon č. 301/2005 Z.z.) v § 90 upravuje povinnost uchování a vydání počítačových dat, v § 91 možnost jejich odnětí. V § 115 odst. 9 pak stanoví, že ustanovení o odposlechu a záznamu telekomunikačního provozu se přiměřeně vztahují též na provozní a obsahové údaje, které jsou v reálné čase přenášeny prostřednictvím počítačového systému. Obdobné ustanovení nalezneme i pokud jde o již uskutečněný telekomunikační provoz (§ 116 odst. 4). Důvodem pro tato zvláštní ustanovení je podle důvodové zprávy[3] i podle teorie[4], že vzhledem na imateriálnost, latentnost a velmi nízkou životnost digitálních stop se nemohou zabezpečovat tradičními zajišťovacími úkony, nebo se nemohou považovat za věci důležité pro trestní řízení.

Jak již bylo naznačeno, současný trestní řád není beze zbytku souladný s požadavky plynoucími z Úmluvy. I když lze výkladem některé nedostatky odstranit, je třeba si uvědomit, že ani použití výkladu či analogie, která je v zásadě v trestním právu procesním přípustná, není bezbřehé. Výkladem nelze rozšiřovat dosah těch ustanovení, kterými se zasahuje do základních práv a svobod osob. Zůstává tak celá řada problémů (např. právo nařídit uchování počítačových dat), k jejichž vyřešení je zapotřebí novelizovat trestní řád.

6 Závazky v oblasti mezinárodní spolupráce

Kapitola třetí Úmluvy obsahuje obecná a konkrétní ustanovení mezinárodní spolupráce, která je koncipována jako **doplňková ke stávajícím nástrojům** (mezinárodním dohodám o mezinárodní spolupráci v trestních věcech, ujednáním dohodnutých na základě jednotných nebo recipročních právních předpisů a vnitrostátních zákonů). Není tedy ambicí Úmluvy nahrazovat nebo stanovit nově a odlišně zásady spolupráce podle stávajících nástrojů.[5] Několik odlišných režimů, které by koexistovaly vedle sebe by mohlo způsobit zmatek a přinést pochybnosti o tom, která ustanovení aplikovat, zda této Úmluvy nebo jiná. Pouze s ohledem na mechanismy obzvláště nezbytné pro rychlou a účinnou spolupráci v oblasti trestné činnosti související s počítači stanoví Úmluva určité požadavky.

Česká republika, resp. ČSFR podepsala a ratifikovala obě hlavní Úmluvy týkající se právní pomoci ve věcech trestních i dodatkový protokol k druhé z nich.

[3] Srov. Parlamentní tisk č. 720, 3. volební období.

[4] Srov. Ivor, J. a kol. *Trestné právo procesné.* Bratislava : Iura Edition, 2006, str. 347.

[5] Např. podle Evropské úmluvy o vzájemné pomoci v trestních věcech (ETS č. 30) a Protokolu k ní (ETS č. 99) či podle dvoustranných smluv.

Konkrétně se jedná o tyto dokumenty:

1. **Evropská Úmluva o vydávání osob**, otevřená k podpisu v Paříži dne 13. prosince 1957 (ETS č. 24); ČSFR podepsala Úmluvu dne 13. února 1992, ratifikovala 15. dubna 1992 a pro ČR vstoupila v platnost 1. ledna 1993.

2. **Evropská Úmluva o vzájemné pomoci v trestních věcech**, otevřená k podpisu ve Štrasburku dne 20. dubna 1959 (ETS č. 30); ČSFR podepsala Úmluvu dne 13. února 1992, ratifikovala 15. dubna 1992 a pro ČR vstoupila v platnost 1. ledna 1993.

3. **Dodatkový protokol k Evropské Úmluvě o vzájemné pomoci v trestních věcech**, otevřený k podpisu ve Štrasburku dne 17. března 1978 (ETS č. 99); Česká republika protokol podepsala dne 18. prosince 1995, ratifikovala 19. listopadu 1996 a vstoupila pro ní v platnost 17. února 1997.

V otázkách neupravených mezinárodními úmluvami, ať univerzálními zmíněnými shora nebo dvou či vícestrannými, se postupuje **subsidiárně podle ustanovení trestního řádu** (§ 375–460y). Není-li právní styk mezi Českou republikou a jiným státem upraven mezinárodní smlouvou, orgány činné v trestním řízení vyhoví žádosti cizího státu jen, **je-li zajištěna vzájemnost**, tj. poskytne-li dožadující stát záruku, že v budoucnu vyhoví obdobné žádosti orgánu České republiky.

Pokud jde o **obsahové vymezení spolupráce**, Úmluva upravuje

a) obecné principy týkající se mezinárodní spolupráce (čl. 23) a vzájemné pomoci (čl. 25, 26),

b) obecné principy týkající se vydávání osob (čl. 24),

c) postupy týkající se žádostí o vzájemnou pomoc v případě neexistence aplikovatelných mezinárodních smluv (čl. 27, 28),

d) konkrétní ustanovení (čl. 29–35) včetně sítě 24/7 (čl. 35).

7 Závěrem

V příspěvku jsem se snažil ve stručnosti podat přehled ustanovení Úmluvy a výběrově též jejich výklad. Závazky z Úmluvy vyplývající konfrontuji s českým právním řádem. Lze konstatovat, že současná právní úprava není souladná s těmito závazky, zejména pokud jde o vyžadovaná procesní opatření. Před ratifikací Úmluvy bude tedy třeba novelizovat příslušné zákony. Bylo by jistě ke škodě, kdyby se Česká republika v důsledku nedostatečné legislativy stala překážkou v boji s kybernetickou kriminalitou na mezinárodní úrovni.

Hierarchická struktura pracoviš? CSIRT a její funkce

Andrea Kropáčová, Robert Malý

E-mail: Andrea.Kropacova@cesnet.cz, Robert.Maly@cz.ness.com

Abstrakt

Součástí preventivní a aktivní ochrany počítačů a počítačových sítí je důsledné a efektivní řešení bezpečnostních incidentů včetně odstraňování jejich příčin a následků. Proto je nutné, aby na možnost narušení bezpečnosti sítě a počítačů byli jejich správci a uživatelé připraveni a měli k dispozici funkční struktury, efektivní postupy, pravidla a technické prostředky vedoucí k co nejrychlejšímu odstranění problémů při minimalizaci škod.

Problematiku řešení bezpečnostních incidentů a jejich prevenci řeší v počítačových sítích obecně tzv. CSIRT týmy – Computer Security Incident Response Team (případně CERT – Computer Emergency Response Team). Existence CSIRT týmu je žádoucí v každé provozované síti, obzvláště pak v těch velkých (universitních, metropolitních) v sítích bank, v sítích velkých poskytovatelů připojení, poskytovatelů obsahu a rizikových služeb. CSIRT dané sítě obecně představuje záchytný bod, na který je možné se obrátit se zjištěným bezpečnostním problémem. Dohromady tvoří CSIRT týmy funkční infrastrukturu, která v prostředí globálního Internetu umožňuje rychle a efektivně zasáhnout v případě závažného bezpečnostního incidentu, případně varovat ostatní potenciální oběti útoku.

Úvod

S narušením bezpečnosti počítačů a sítí se Internet setkává prakticky od počátku svého vzniku. Částečně je to způsobeno tím, že na počátku nebyl koncipován tak, aby bezpečný byl. Měl sloužit na testování a ověřování vědeckých teorií. Proto nikoho v 70. letech, kdy začaly protokoly TCP/IP vznikat, nenapadlo se zabývat také bezpečností. Důraz byl kladen na robustnost a efektivitu, které by zaručily chod sítě v případě válečného konfliktu a zničení některé její části. První veřejně známý mezinárodní bezpečnostní incident v ARPANET (Internet)

se objevil v roce 1986 (program neoprávněně kopírující informace z počítače). Postiženy byly počítače v několika zemích a to nejen v akademické sféře, ale i ve státní správě a v silových složkách, jako je např. armáda. Po tomto incidentu došlo k prvnímu uvědomění si, že ARPANET může být použit i k destruktivním účelům. Od počátku 90. let, kdy se Internet dostal k masám a to především prostřednictvím škol k mladým lidem, jsou bezpečnostní problémy počítačů a sítí na denním pořádku. Slovo *hacker*, které se na svém začátku používalo pro odborníky pronikající do tajů SW, postupně ztratilo svůj původní význam a dnes je používáno téměř výhradně jako označení pro člověka, který cíleně narušuje bezpečnost počítačů a služeb..

Bezpečnostní incidenty

Uživatelé často kladou otázku, proč vlastně existují snahy nabourat bezpečnost počítačů, sítí a služeb? Odpověď je neradostná, ale prostá. Samozřejmě jde v první řadě o peníze. Většina útoků je zaměřena na získání privátních informací uživatele, které by vedly k prozrazení např. přístupových kódů k bankovnímu účtu (stále populárnější problém zvaný *phishing*), získání privátních firemních informací (smlouvy s klienty, kontakty, know-how), které by bylo možné zpeněžit a vůbec jinak zneužít. Za snahou obohatit se nezaostávají ani snahy někomu ublížit a uškodit mu třeba na poli soukromého partnerského života, nebo v zaměstnání.

Velkým lákadlem pro narušitele bezpečnosti jsou sítě s dobrou konektivitou a s velkým množstvím výkonných počítačů, s velkými a zajímavými zdroji dat a podobně. Tam, kde není možné ukrást peníze, je stále ještě možnost zneužít stroje jako přestupní stanice pro páchání dalších pirátských činů, např. *spammingu* (nevyžádaná pošta), porušování autorských práv vystavením dat chráněných autorským zákonem na napadeném serveru s velkou diskovou kapacitou, narušení integrity osobních dat apod.

Obecně lze bezpečnostní incident definovat jako narušení bezpečnosti IS/IT a pravidel stanovených k jeho ochraně (**bezpečnostní politika**). I přes maximální snahu na poli prevence a péče o síť a služby existuje reálná možnost, že dojde k narušení bezpečnosti počítače, správce tento problém nezaregistruje včas, a stroj v jeho správě je napaden a zneužit k dalším útokům.

V takovém případě se ovšem správci napadených strojů brání a na útok si stěžují. Základním komunikačním prostředkem je v těchto případech **elektronická pošta**. Na kontaktní e-mail adresy správců zodpovědných za konkrétní síť jsou v případech bezpečnostních incidentů posílána tzv. **hlášení o incidentu**.

Při zjištění vzniklého bezpečnostního incidentu je nejdůležitější **rychlá a adekvátní reakce**, což zahrnuje mít připravené postupy řešení jednotlivých bezpečnostních incidentů (v okamžiku zjištění BI není čas na zjišťování jak s inci-

dentem naložit, je čas na spuštění předem připraveného plánu obrany a obnovy), mít v záloze základní disaster-recovery plán, plán zvážení rizik a podobně.

Co je CSIRT/CERT tým?

Zkratka CSIRT znamená „Computer Security Incident Response Team“, tedy „tým, který zodpovídá za řešení počítačových bezpečnostních incidentů“. Ve stejné souvislosti a stejně hojně se používá i termín CERT (Computer Emergency Response Team). První CERT tým vznikl v roce 1988 na Carnegie Mellon University jako reakce na tzv. *Morissův červ*, který tehdy v plné nahotě ukázal zranitelnost Internetu, ale především jeho nepřipravenost. Nakonec nejcennějším výsledkem práce tohoto týmu, který byl zřízen za účelem nalezení účinné obrany proti jedné události, bylo poznání, že nejefektivnější model je být na možnost podobných narušení bezpečnosti předem připraven. Zkratku CERT si Carnegie Mellon nechala registrovat jako ochrannou známku a ač se jejímu užití v tomto kontextu nebrání a podporuje ji, přece jen to bylo příčinou vzniku a zavedení druhého pojmu – CSIRT.

Obecně lze říct, že CERT/CSIRT tým je tým, který poskytuje podporu a služby v oblasti bezpečnosti počítačových sítí a služeb na ní provozovaných, a to především v oblasti ***řešení bezpečnostních incidentů*** (*incident response*). Je to místo, ve kterém se shromažďuje potřebné know-how a zkušenosti v oblasti boje proti počítačové kriminalitě a řešení BI tak, aby následky byly minimální. V kontextu globálního Internetu je pak možné CERT/CSIRT týmy vnímat jako infrastrukturu, která umožňuje rychlou komunikaci, efektivní reakci a prevenci. Při práci CSIRT čerpá především ze svých zkušeností, předem připravených a v praxi ověřených postupů a ze spolupráce s ostatními CSIRT týmy.

Co dělá CSIRT CSIRTem?

Ne každý tým se z ničeho nic může prohlásit za CERT/CSIRT tým. Přesněji – větší zádrhel je v dosažení stavu, kdy jej již existující CERT/CSIRT týmy jako takový akceptují. Nicméně cesta k získání statutu CERT/CSIRT tým není složitá, na jejím začátku stačí jasným způsobem deklarovat přibližně následující:

- pole působnosti týmu (za co tým zodpovídá),
- nabízené služby,
- základní kontaktní informace – členové týmu, organizace provozující tým, e-mail adresa pro komunikaci, telefonní číslo, adresa apod.

Pole působnosti týmu, aneb za co tým zodpovídá, se samozřejmě odvíjí od toho, o jaký tým jde. V zásadě je možné zřídit týmy těchto typů:

- ***interní*** – slouží a zodpovídá za konkrétní síť (např. za konkrétní rozsah IP adres, které patří do dané sítě, domény), je zřízen provozovatelem sítě a ten mu na počátku stanoví jeho pravidla a zodpovědnost,
- ***koordinační*** – tým, jehož hlavním úkolem je koordinovat řešení bezpečnostních incidentů, nemusí je cíleně řešit,
- ***vendor*** – tým zabývající se řešením bezpečnostních incidentů, které se dotýkají kokrétního produktu (SW),
- ***národní, vládní*** – speciální případy založené na principech prvních dvou zmíněných týmů, jejich pole působnosti do jisté míry záleží na zřizovateli a legislativě země.

Služby nabízené týmem mohou zahrnovat následující – školení, varování před aktuálními útoky, slabinami OS, bezpečnostní audity, SW konzultace, bezpečné konfigurace, vývoj a provoz nástrojů pro sledování provozu sítě a služeb a mnoho dalších. Minimem je ovšem řešení bezpečnostních incidentů.

V okamžiku, kdy se nově zřízený CSIRT/CERT tým vypořádá s výše uvedenými body a stanoví si základní týmovou politiku *incident handlingu* (*politika řešení bezpečnostních incidentů*), *která obnáší* klasifikaci vážnosti incidentů, pravidla reakce na incidenty, dosažitelnost členů týmu, pravidla pro komunikaci s autorem stížnosti apod. apod., je na dobré cestě, aby jej okolní týmy akceptovaly. Samozřejmou a nezbytnou součástí je nutnost seznámit se a dodržovat základní pravidla, na kterých se CSIRT komunita dohodla.

Čím je zaručena důvěryhodnost CSIRT/CERT týmu?

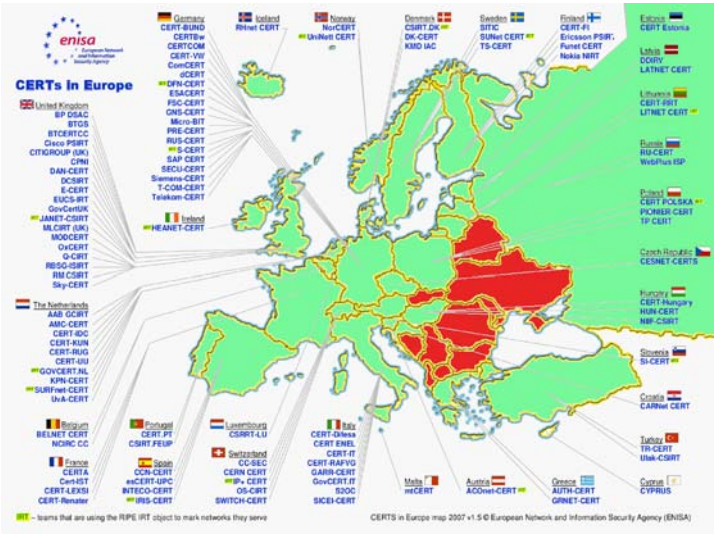
Ve světě CSIRT/CERT týmů a při jejich komunikaci je kladen velký důraz na důvěru, osobní kontakty a vazby a vzájemnou spolupráci. Vhodné platformy, které se touto problematikou zabývají, jsou nadnárodní organizace TERENA (Trans-European Research and Education Networking Association), FIRST (Forum of Incident Response and Security Teams) a ENISA (European Network and Information Security Agency). Všechny tyto platformy pracují na vývoji standardů, pravidel a doporučení, organizují osvětové akce (školení, tréninky), ale také umožňují pravidelná setkávání členů bezpečnostních týmů. V rámci organizace TERENA je možné se zapojit do aktivity TF-CSIRT (Task Force for CSIRT) a zúčastňovat se pravidelných setkání, která se konají třikrát ročně.

Organizace FIRST kromě velké výroční konference (letos se koná v červnu ve Vancouveru, Kanada) pořádá řadu školení a spolupracuje s TF-CSIRT. Dalšími zajímavými aktivitami, které stojí za zmínku, jsou aktivity „CSIRT Training“ a „Trusted Introducer“. CSIRT Trainng slouží pro vyškolení nových členů CSIRT/CERT týmů a jsou dobrým startem pro založení týmu tohoto typu.

Platformy TERENA a FIRST navíc plní roli jakési „záruky“, že tým, který o sobě tvrdí, že je CSIRT/CERT týmem, jím opravdu je a že jím deklarovaný model chování je pravdivý. V případě organizace TERENA se tohoto stavu dosahuje tzv. „akreditačním“ procesem v rámci TERENA TI (Trusted Introducer), v případě organizace FIRST se dosahuje statutu „člena“. Oba procesy mají jedno společné – jde o zjištění maximálního množství informací o daném týmu, popisu jeho chování a vnímání problematiky řešení bezpečnostních incidentů tak, aby to korespondovalo s požadavky komunity.

Organizace FIRST má v současné době (duben 2008) 193 členů ze všech zemí světa, přičemž v Evropě je registrováno zhruba 100 týmů typu CSIRT/CERT. V České republice existuje v současné době pouze jeden oficiální CSIRT tým, který spolupracuje s výše uvedenými organizacemi, a to tým nazývaný CESNET-CERTS, který provozuje sdružení CESNET, z. s. p. o. (http://www.cesnet.cz).

Uvedený obrázek ilustruje stav CSIRT/CERT aktivit v evropských zemích. Země, ve kterých neexistuje ani jeden oficiální CSIRT/CERT tým, jsou vyznačeny červenou barvou, země, ve kterých je evidován alespoň jeden tým, jsou vybarveny zeleně. Autorem obrázku je organizace ENISA a jeho originální podoba je k nalezení na této adrese:
http://enisa.europa.eu/cert_inventory/downloads/CERT_Euromap.pdf.



CESNET-CERTS

Bezpečnostní tým CESNET-CERTS byl zřízen a je provozován sdružením CESNET, z. s. p. o. a jeho hlavním úkolem je řešení a koordinace řešení bezpečnostních incidentů v síti národního výzkumu CESNET2. Tým CESNET-CERTS začal vznikat na přelomu let 2003–2004 a oficiálně byl ustanoven v lednu 2004, kdy se dostal do stavu „listed“ v pojetí TERENA TI a ocitl se tak na seznamu oficiálních evropských CSIRT týmů (http://www.ti.terena.org/teams/). Na jeho počátku stáli tři zaměstnanci sdružení CESNET. Dnes, v roce 2008, má tým členů pět. Od roku 2006 část rutinního provozu týmu zajišťují členové *Pracoviště stálé služby*, což je pracoviště určené pro stálý (365/7/24) dohled nad sítí CESNET2.

Oblasti zájmu a služby, které tým CESNET-CERT nabízí, jsou:

- příjem hlášení bezpečnostních incidentů vzniklých v síti CESNET2 a jejich řešení,
- provoz a rozvoj IDS (Intrusion Detection System),
- edukace – školení, publikace.

Z hlediska zásad o definování a fungování týmu typu CSIRT/CERT, které jsme si v tomto článku už zmínili, je možné popsat tým CESNET-CERTS následujícím způsobem:

- je provozován sdružením CESNET, z. s. p. o.,
- základní informace jsou na http://www.cesnet.cz/csirt/,
- má 4 členy,
- zajišťuje příjem hlášení bezpečnostních incidentů a koordinuje řešení bezpečnostních incidentů v síti národního výzkumu CESNET2, to znamená:
  - v těchto adresových blocích, které jsou součástí AS2852[1] – 78.128.128.0/17, 146.102.0.0/16, 147.32.0.0/15, 147.228.0.0/14, 147.251.0.0/16, 158.194.0.0/16, 158.196.0.0/16, 160.216.0.0/15, 193.84.32.0/20, 193.84.160.0/20, 193.84.192.0/19, 195.113.0.0/16, 195.178.64.0/19,
  - v doménách – cesnet.cz, cesnet.eu, cesnet-ca.cz, liberouter.net, liberouter.eu, flowmon.eu, flowmon.org, netopeer.org, ten.cz, acad.cz, ipv6.cz, cesnet2.cz, eduroam.cz, medigrid.cz, cifer.cz, ces.net, liberouter.org,

[1] AS = Autonomous System (Autonomní systém)

- přijímá hlášení bezpečnostních incidentů na adrese abuse@cesnet.cz a certs@cesnet.cz a přijímá i adresy typu hostmaster@ a postmaster@ svých jmenných domén,
- členové týmu odpovídají z adresy `certs@cesnet.cz` a odchozí zprávy podepisují PGP klíčem,
- PGP klíč CESNET-CERTS:
  - User ID: CESNET-CERTS `<certs@cesnet.cz>`
  - Key ID: 0x9CAA8579
  - Key type: DH/DSS
  - Key size: 1024 bits
  - Expiration: never
  - Fingerprint: 341D 3EB0 0160 941F 6A06 4401 F9BF C741 9CAA 8579
- Další kontaktní informace jsou k nalezení na stránkách provozovatele týmu (http://www.cesnet.cz/).

Členové bezpečnostního týmu CESNET-CERTS se od roku 2004 zúčastňují pravidelných setkání TF-CSIRT, konferencí a školení FIRST a dalších podobně zaměřených akcí. V květnu roku 2007 sdružení CESNET hostovalo 21. zasedání TF-CSIRT.

NATO CIRC

NATO CIRC (Computer Incident Response Capability) je aktivita v rámci NATO, vycházející ze závazku členských zemí NATO k projektu NATO G 2868, navazující na předchozí aktivity NATO v rámci NATO COMPUTER INCIDENT RESPONSE CAPABILITY koncepčně „nastartované“ v roce 2002.

Samotná idea NATO CIRC vychází a dodržuje principy fungování CSIRT ve světě. Fakticky se jedná o vybudování hierarchie CIRC středisek v rámci NATO, kde vrcholem hierarchie je NCIRC CC v Belgii (Koordinační centrum NATO CIRC) http://www.ncirc.nato.int/. NCIRC CC je v současné době s komunitou CSIRT ve světě spojeno jak na evropské scéně – TERENA TI tak i celosvětově – FIRST. První prezentace aktivity NATO CIRC vůči evropské bezpečnostní komunitě proběhla na TF-CSIRT meeting v roce 2004
http://www.terena.org/activities/tf-csirt/meeting11/NCIRC-Anil.pdf.

Jednotlivé členské země NATO si vybudovali svá CIRC pracoviště jejichž základním přínosem je dosáhnout schopnosti centrálně sledovat, vyhodnocovat

a následně reagovat na bezpečnostní incidenty. Cílem není poskytnout pouze možnost operativní reakce na incidenty, ale také poskytnout nástroj pro dlouhodobé sledování trendů.

CSIRT.CZ

Vybudování a provoz pracoviště CSIRT.CZ je jedním z dílčích úkolů projektu *„Kybernetické hrozby z hlediska bezpečnostních zájmů České republiky“*, který byl vypsán a je financován Ministerstvem vnitra České republiky. S budováním týmu CSIRT.CZ se začalo v létě v roce 2007 a dne 3. dubna 2008 byl spuštěn jeho pilotní provoz. Zakládajícími členy týmu CSIRT.CZ jsou členové bezpečnostního týmu CESNET-CERTS. Členové CESNET-CERTS zajišťují v prostředí CSIRT.CZ jeho rutinní funkce, ale především zde fungují jako zdroj know-how a praktických zkušeností, které jsou pro chod CSIRT týmu potřeba a jako školitelé nových členů.

Hlavním úkolem CSIRT.CZ je koordinace a pomoc při řešení bezpečnostních incidentů, které mají původ v sítích provozovaných v České republice a na jejichž oznámení správci dané sítě nereagují nebo takovou osobu není možné dohledat nebo se vyskytnou jiné problémy, např. jazykové. Obecně se ale z pohledu stěžovatele jedná o kombinaci „vážný a neřešený bezpečnostní incident“. V takovém případě vlastně CSIRT.CZ funguje jako „místo poslední záchrany“, na které je možné se obrátit se žádostí o pomoc. V žádném případě ale CSIRT.CZ nesmí být chápáno jako centrum, kam se budou obracet uživatelé s problémy typu „něco divného se mi děje s počítačem“ nebo „někdo mi na mém webhostingu smazal moje WWW stránky“ nebo „ve školní laboratoři je PC a to je zjevně hacknuté, co s tím?“ apod. Uživatelé by měli vědět a být poučeni o základních principech hlášení BI a především pravidlech této problematiky v sítích, ve kterých se vyskytují (školní, firemní) a incidenty hlásit lokálnímu správci.

V prostředí „českého“ Internetu jsou cíle CSIRT.CZ následující: kromě provozu „místa poslední záchrany“ v oblasti řešení bezpečnostních útoků, což je jádro každého týmu CSIRT/CERT, jde především o to motivovat provozovatele velkých sítí (např. poskytovatele Internetu) a rizikových služeb (např. banky provozující elektronické bankovnictví) ke zřízení oficiálních týmů CSIRT/CERT a o spolupráci mezi nimi. CSIRT.CZ má ambici sloužit jako modelové řešení, zdroj know-how, zkušeností a platforma pro diskusi a spolupráci.

Literatura

[CSIRT.CZ] http://www.csirt.cz/

[CESNET, z. s. p. o.] http://www.cesnet.cz/

[CESNET-CERTS] http://www.cesnet.cz/csirt/

[TERENA] http://www.terena.org/

[TERENA TI] http://www.ti.terena.nl/

[TF-CSIRT] http://www.terena.org/activities/tf-csirt/

[FIRST] http://www.first.org/

[ENISA] http://www.enisa.eu/

[US CERT] http://www.cert.org/

[NATO CIRC] http://www.ncirc.nato.int/

Forenzní analyzátor pro operativní analýzu

Róbert Lórencz, Tomáš Zahradnický, Jiří Buček

E-mail: lorencz@fel.cvut.cz, zahradt@fel.cvut.cz, bucekj@fel.cvut.cz

Abstrakt

V rámci projektu „Problematika kybernetických hrozeb z hlediska bezpečnostních zájmů České republiky“ je na Katedře počítačů FEL ČVUT v Praze vyvíjen forenzní analyzátor osobních počítačů. Vyvíjený forenzní analyzátor je určen zejména pro nejnižší úroveň nasazení – operativní analýzu – při odhalování počítačové kriminality vyškolenými vyšetřovateli. Při vývoji forenzního analyzátoru je kladen důraz na jeho jednoduchost, přenositelnost a otevřenost kódu. Cílem tohoto příspěvku je ve zkratce představit východiska, požadavky plynoucí z praxe a navržené technické řešení forenzního kitu.

Klíčová slova: Forenzní analýza, operativní analýza, digitální stopa

1 Úvod

Forenzní analyzátory jsou používány pro získávání dat a dalšího důkazního materiálu – digitálních stop – ze záznamových paměťových médií počítačových systémů a to zejména z jejich pevných disků. Forenzní analyzátory lze rozdělit na hardwarové a softwarové. Hardwarové analyzátory zjišťují data přímo z pevného disku obvykle na nejnižší analogové úrovni. Mají tak vyšší rozlišovací schopnost, avšak za vyšší pořizovací cenu. Na druhé straně softwarové analyzátory se obvykle používají na speciálně upraveném počítači, ke kterému se připojí pevný disk přes rozhraní, které blokuje zápis na analyzovaný disk na hardwarové úrovni případně na softwarové úrovni a to za účelem zamezení nežádoucí změny obsahu zkoumaného disku. Druhou možností je provádět forenzní analýzu na kopii zkoumaného disku, ale i v tomto případě se požaduje neinvazivní přístup. Typickým příkladem softwarového forenzního analyzátoru je produkt enCase Forensic Edition americké společnosti Guidance Software [4].

Určitou nevýhodou informací získaných například prostřednictvím profesionálních analyzátorů je skutečnost, že nemáme k dispozici zdrojové kódy analyzátoru. Proces použití analyzátoru (jeho softwaru) při pořizování digitálních stop je netransparentní a do určité míry neovlivnitelný. Je proto užitečné mít k dispozici zdrojový kód analyzátoru, a tak zprůhlednit způsob získávání informací prostřednictvím analyzátoru, případně umožnit jeho vývoj a modifikaci.

Další nevýhoda profesionálních forenzních analyzátorů je jejich složité ovládání s množstvím různých funkcí, které jsou obvykle nepříliš používané. Při provádění prvotní, rychlé, operativní analýzy je takto komplexní systém spíš nevýhodou, a to i s ohledem na odlišnou úroveň vyškolených expertů provádějících tuto analýzu. Cena profesionálních analyzátorů je nemalá a pokud nejsou tyto analyzátory plně využívány, a v operativní analýze ani být nemohou, může být jejich pořizování ekonomicky málo efektivní. Určitým nedostatkem profesionálních analyzátorů je absence podpory českého jazyka jak při komunikaci s analyzátorem, tak také v procesu samotné forenzní analýzy.

Vzhledem k uvedeným nedostatkům profesionálních forenzních analyzátorů vyvstal požadavek na základě zkušenosti policejních znalců vytvořit forenzní analyzátor přizpůsobený podmínkám operativní analýzy s otevřeným kodem, s vybraným malým množstvím nejužitečnějších funkcí a podporou češtiny. Na Katedře počítačů FEL ČVUT v Praze je týmem složeným z pedagogů, vědeckovýzkumných pracovníků a studentů vyvíjen v rámci projektu „Problematika kybernetických hrozeb z hlediska bezpečnostních zájmů České republiky“ forenzní analyzátor počítačových systémů splňující tyto požadavky. Při vývoji forenzního analyzátoru je kladen důraz na přenositelnost, jednoduchost a otevřenost kódu. Cílem tohoto příspěvku je představit vyvíjený **Operativní Forenzní Analyzátor (OFA)**, uvést východiska a požadavky plynoucí z praxe, které vedly k jeho návrhu a popsat navržené technické řešení.

2 Požadavky kladené na OFA

Existuje množtví definic forenzní analýzy digitálních stop. Ve zkratce lze říci, že forenzní analýza počítačových systémů využívá transparetních metod informatických vědních oborů ke sběru, identifikaci, analýze, interpretaci, dokumentaci a prezentaci digitálních stop ze zdrojů digitálních nosičů dat. Tato činnost je prováděna s cílem rekonstruovat události, které by mohly být shledány jako zločinné nebo by mohly vést k odhalení neautorizovaných akcí, které působí rušivě až destruktivně na plánovaný běh operací. Jednotlivé fáze forenzní analýzy jsou: zajištění a analýza objektů dat ke zkoumání, a vytvoření výstupu požadovaných informací plynoucích z předchozích dvou kroků.

Nejčastějšími zkoumanými objekty u počítačových systémů jsou záznamová paměťová média (pevné disky, paměťové karty, CDR, CD-RW, DVD-R, DVD-

RW, PCMCIA, atd.) Na těchto paměťových médiích jsou pomocí forenzní analýzy zkoumány a zajišťovány digitální stopy, tj. jakékoliv informace s vypovídající hodnotou související se zkoumaným případem.

Výstupem forenzní analýzy je znalecký posudek – zpráva. Znalecký posudek by měl obsahovat kromě samotné zprávy týkající se zkoumaného případu také písemný popis všech úkonů k zajištění digitálnich stop a použití všech softwarových a hardwarových nástrojů.

V současné době musí vyšetřovatel příliš spoléhat na forenzního znalce, aby byl schopen získat data, která potřebuje pro vyšetřování. Nástroje, které jsou mu k dispozici, jsou orientovány na uživatele-experty, kterým ale vyšetřovatel být nemusí. To je další z důvodů proč realizovat forenzní analyzátor pro operativní analýzu – OFA. V případě návrhu OFA je velmi důležitý výběr funkcí analyzátoru. Důraz je kladen na jednoduchost ovládání, jednoduchost a transparentnost kodů a hlavně na správný a z hlediska vyšetřovatelů na co nejucelenejší výber funkčních možností pro provedení hodnotné a věrohodné analýzy dat. Výběr jednotlivých funkcí OFA a samotné jeho provedení je uvedeno v dalších částech.

3 Charakteristika OFA

OFA je nástroj, který má pomoci vyšetřovateli ve vyhledání digitálních stop, které by přinesly důležité informace do procesu vyšetřování. Jeho výstupy nejsou přímo důkazy, které by byly předloženy soudu, proto není nutné protokolovat všechny akce, ani zajišťovat zkoumaná data proti přepsání.

Vstupem pro operativní analýzu je kopie dat, kterou vyšetřovateli poskytne forenzní znalec nebo kriminální technik. V procesu analýzy vyšetřovatel pomocí OFA vyhledá v datech na základě okolností zkoumaného případu soubory, které jsou pro daný případ relevantní, k čemuž mu slouží jednak automatické hledání na základě zadaných kritérií, a také manuální procházení nalezených souborů v zabudovaných prohlížečích. Výstupem OFA je jednak kopie zajímavých souborů pro použití v dalším vyšetřování, jednak zpráva (report) pro evidenci a jako podklad pro forenzního znalce pro vyhotovení znaleckého posudku pro soud.

3.1 Spustitelnost

OFA by měl být spustitelný na co nejširším okruhu počítačů a neměl by být závislý na operačním systému, který je na tom kterém počítači nainstalován. Jednou z možností jak toho dosáhnout je vytvořit spustitelné (bootovatelné) CD, které spustí svůj vlastní operační systém (např. Linux, FreeBSD apod.). Výhoda takového řešení je v nezávislosti na softwaru nainstalovaném na použitém počítači, čímž odpadá velká část testování kompatibility, schvalování instalace na počítač a podobně.

Dalším důvodem pro spustitelné CD je také kontrolované odpojení počítače od sítě a všech ostatních připojení do doby, než se sám uživatel OFA rozhodne některé takové připojení specificky povolit. Síťová připojení je možno zprovoznit např. za účelem uložení (vykopírování) vybraných souborů, uložení zprávy nebo vytvoření většího odkládacího prostoru.

Kromě spouštění z CD přichází v úvahu i spouštění z jiných médií, např. z DVD, USB FLASH apod.

3.2 Specifikace úkonu

Na začátku procesu operativní analýzy předloženého média je specifikace vstupu, tj. určení, kde se nachází analyzovaná data. To může být buď na externím médiu (pevném disku, CD, DVD, flash, ...) nebo pevném disku přímo v použitém počítači. Analyzované médium může být buď přímo kopie (jedna ku jedné), nebo obraz média v souboru, případně jen adresářová struktura s kopií původních souborů. OFA musí umět všechny tyto typy vstupů zpracovat.

Pro usnadnění evidence případných zjištění je vhodné umožnit zadání údajů jako je číslo jednací, jméno uživatele, aktuální datum a čas (automaticky), a případné poznámky.

V průběhu analýzy některých souborů (zejména archivů) je potenciální potřeba většího množství paměti. Je tedy vhodné, aby program umožňoval použití odkládacího prostoru na zvoleném pomocném médiu, a na začátku umožnil příslušnou volbu.

3.3 Hledání

Jádrem procesu analýzy je prohledávání datového média podle zadaných kritérií. Kromě obvyklých kritérií jako je přípona souboru, čas modifikace, velikost souboru a klíčová slova v souboru má program umět i analýzu typu souboru podle jeho obsahu (jako příkaz `file` v UNIXu).

Program by měl umět vyhledávat i ve smazaných souborech, a to do té míry, aby i uživatel neznalý přesné vnitřní struktury toho kterého souborového systému uměl případný nález zpracovat. Je tedy vhodné provést odhad, do jaké míty je smazaný soubor ještě obnovitelný, ale neprezentovat uživateli příliš mnoho technických detailů, které by ocenil jen znalecký expert (který má ovšem dostatek jiných nástrojů pro takovou analýzu).

Pro rychlé a přehledné zadání vyhledávacích kritérií je nutné poskytnout přednastavené kategorie jako např. Dokumenty, Obrázky, Videa apod., aby uživatel nemusel zadávat mnoho konkrétních typů souborů (JPEG, PNG, GIF).

3.4 Prohlížení

Po vyhledání souborů odpovídajících zadaným kritériím je uživateli prezentován výsledek v takové formě, aby měl možnost jednotlivé soubory prohlédnout. Seznam souborů je roztříděn do kategorií podle typu, zároveň je k dispozici i informace o umístění souborů.

Pokud to jde, měl by být zobrazen náhled obsahu souboru, a to jak zmenšený (zobrazený spolu s náhledy ostatních souborů), tak zvětšený na plnou velikost. OFA samozřejmě nebude umět zobrazit všechny typy souborů, potom je potřeba alespoň dát informaci, o jaký typ souboru se jedná, a nejlépe i navrhnout prohlížeč, kterým by se dal daný soubor zobrazit.

Uživatel by měl mít možnost každý soubor označit jako zajímavý (pro pozdější export) a případně k souboru napsat vlastní poznámku.

3.5 Export souborů a zprávy (reportu)

Označené soubory budou exportovány na počítač vyšetřovatele nebo na vybrané externí médium (jiné než analyzované). Vzhledem k tomu, že vybrané soubory se mohly na analyzovaném médiu nacházet na různých místech adresářové struktury, přicházejí v úvahu dva režimy exportu. Buď se soubory uloží do kopie původního adresářového stromu, nebo se exportují do jednoho adresáře s tím, že případné konflikty názvů souborů budou nějakým způsobem vyřešeny (např. číslováním).

Kromě vlastních souborů je potřeba uložit také zprávu (report) o nalezených souborech. Zpráva bude obsahovat jména exportovaných souborů i s cestou a informaci o typu a případně s poznámkou vyšetřovatele. Zpráva by měla být snadno dále zpracovatelná např. importem do tabulkového procesoru, takže je vhodné ji generovat jako tabelovaný textový soubor.

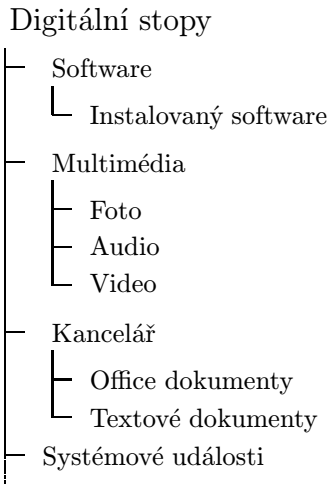
Zpráva je podkladem pro forenzního znalce, který má přístup k původnímu (zabavenému) datovému médiu nebo jeho certifikované kopii. Forenzní znalec poté na základě informace ze zprávy od vyšetřovatele může vyhotovit znalecký posudek a vytvořit certifikovanou kopii původních dat pro účely soudního řízení.

4 Uživatelské rozhraní

OFA, jak již bylo konstatováno, musí být zejména jednoduchý a musí být použitelný širokým spektrem uživatelů, což běžně používané forenzní analyzátory [4, 5, 6] nesplňují zejména po stránce náročnosti na uživatele. Navrhovaný OFA předpokládá pouze poučeného uživatele, který nemusí mít žádné znalosti o způsobech uložení dat v souborových systémech, formátech souborů, počítačovém hardware nebo znalosti interakce s příkazovou řádkou. Na cílového uživatele nejsou kladeny ani žádné jazykové znalosti, protože cílem projektu vývoje OFA

je uživateli poskytnout takový nástroj, který bude s uživatelem komunikovat česky a skryje maximum systémově závislých aspektů.

Uživatel dokonce ani nemusí rozumět souborovému systému na zkoumaném médiu vzhledem k tomu, že u OFA je upřednostňováno zobrazení souborového systému podle typu dat, který uživatel vidí ve formě stromu digitálních stop, nikoliv adresářové struktury. Strom digitálních stop je zobrazen na obrázku 1.



Obr. 1: Strom digitálních stop

Zobrazení ve formě adresářové struktury je stále možné, avšak vyžaduje sžití vyšetřovatele s myšlenkovými pochody tvůrce souborů, který se navíc mohl pokoušet souborům změnit typ, například změnou přípony souboru, případně umístit soubory například mezi systémové soubory, a takto se pokusit vyšetřovatele zmást. Strom digitálních stop kategorizuje soubory podle jejich skutečného typu, který se neurčuje podle přípony, ale podle obsahu souborů pomocí nástroje file-4.23. S použitím databáze známých hashí [3] může OFA vyloučit z dalšího hledání a zpracování soubory, které jsou součástí systému, případně jiných známých aplikací, a nenesou tak žádnou užitečnou informaci.

5 Implementace OFA

Určitou inspirací pro návrh OFA byl forenzní analyzátor vytvořený na MFF UK [1]. Analýza kódu FA psaného na MFF UK ukázala, že jazyk Java je nevyhovující. Proto pro realizaci FA byl zvolen programování jazyk C/C++ a to

převážně z důvodů kompatibility s OpenSource softwarem, a dále také kvůli možnostem adaptovat moduly pro souborové systémy přímo z operačních systémů s volně dostupným zdrojovým kódem. Pro grafické rozhraní byla zvolena knihovna Qt společnosti TrollTech [2], která je otevřená a zdarma pro vývoj softwaru pod licencí GPL. OFA používá sadu dalších knihoven pro zobrazování náhledů souborů, práci s daty, atd. Pro zobrazování náhledů dat je vhodná komerční knihovna RasterMaster Imaging SDK for Windows a RasterMaster Imaging SDK for UNIX [7], protože je schopna zobrazit dokumenty mnoha formátů, které nejsou dokumentovány[1]. I přes její vysokou cenu je tato knihovna výhodná, protože možnost instantního náhledu téměř jakéhokoliv statického obrázku nebo dokumentu umožňuje rychlou orientaci v datech. Po stránce náhledů filmů je plánováno začlenění přehrávače *mplayer*, rozšířeného o přehrávání dalších typů souborů (3GPP, AAC, AMR).

Aktuálně má OFA podobu spustitelného CD na bázi Knoppix Linuxu. Tento operační systém byl zvolen díky tomu, že jsou k dispozici jeho zdrojové kódy a máme možnost do systému zasáhnout na kterékoliv úrovni. To nám umožní například zakázat všechna síťová rozhraní, připojovat disky a diskové obrazy v režimu pouze pro čtení, atd. Díky své modularitě a snadné konfiguraci distribučních CD byl vybrán Knoppix Linux s jádrem 2.6. Analyzátor může běžet v případě potřeby i na operačním systému Microsoft Windows, protože knihovna Qt, náhledová knihovna a další jeho součásti jsou implementovány i pro Windows. Část OFA je samozřejmě závislá na použité platformě. Zdálo by se, že se může vyskytnout problém s přímým přístupem na disk pomocí modulů pro souborové systémy, ale tento problém neexistuje díky vrstvě abstrakce pro komunikaci s diskem, která je pod moduly souborových systémů. Ta existuje v implementaci pro Linux a pro Windows a platformově rozdílné prvky jsou tak odstíněny. OA může zkoumat adresář na disku, obraz disku, anebo přímo některé z připojených médií/disků na nízké úrovni. Pro tyto účely obsahuje OA upravené verze souborových systémů, které poskytují rozšířenou funkcionalitu oproti normálním modulům pro souborové systémy.

6 Závěr

Vyvíjený operativní forenzní analyzátor – OFA pro operativní analýzu počítačových systémů má několik specifických vlastností, které ho odlišují od komerčně dostupných analyzátorů. Je především určen pro nasazení v „první vlně“ pro policejní vyšetřovatele, kteří nemusejí být forenzními znalci. Naším cílem je vytvořit OFA spustitelný v různých systémových prostředích, s jednoduchým ovládáním a s komunikací v češtině. Jeho další vlastnosti jsou přenositelnost a otevřenost

[1] Mezi tyto formáty řadíme například dokumenty Microsoft Word, Microsoft Excel, Microsoft PowerPoint a některé další.

kódu. Výstup z tohoto OFA má především sloužit pro zajištění digitálních stop pro další zkoumání prováděné zkušenými experty na profesionálních analyzátorech.

Literatura

[1] MFF UK: *Forenzní analyzátor*, studentský projekt forezního analyzátoru vyvíjený v Javě. http://urtax.ms.mff.cuni.cz/forensicanalyzer/

[2] Troll Tech: *Knihovna Qt*. http://www.trolltech.com/

[3] Information Technology Laboratory, National Software Reference Library: *Reference Data Set*. http://www.nsrl.nist.gov

[4] Guidance Software, Inc: *enCase Forensic Edition v6*, 2008. http://www.guidancesoftware.com

[5] *The Sleuth Kit*. http://www.sleuthkit.org

[6] U.S. Internal Revenue Service: *iLook Investigator*. http://www.ilook-forensics.org/

[7] Snowbound Software Corp.: *RasterMaster Imaging SDK for UNIX and Windows*. http://www.snowbound.com

Skenování otevřených zdrojů

Leo Galamboš

E-mail: leo.galambos@mff.cuni.cz

Klíčová slova: Robot, crawler, Web

Abstrakt

Efektivní skenování (crawling) otevřených zdrojů, především webových dokumentů, je samostatnou disciplínou, ve které se mísí několik dílčích úloh. Mezi ně patří plánování postupu skenování s ohledem na aktuální dostupnost zdrojů, předvídání nejrozličnějších pastí, a v neposlední řadě též využití vhodných datových struktur pro podporu dostatečného výkonu. Článek obsahuje popis konkrétní implementace robota v systému Egothor2, včetně několika výstupů z procházení webových serverů v doméně CZ.

1 Úvod

V roce 2003 byla publikována studie [7], podle které obsahoval tehdejší web až 170 TB dat. Je zřejmé, že od té doby prošel – jako jeden z nejpopulárnějších otevřených zdrojů současnosti – bouřlivým vývojem. Ten byl doprovázen zavedením multimediálních služeb (YouTube *2005), ale i vznikem celé řady nových webových frameworků (Django *2005), které napomohly přenosu mnoha dalších služeb na webovou platformu. První faktor zapříčinil růst webu co do jeho objemu, druhý z faktorů naopak zvýšení počtu dostupných URL. Úkolem skenování je pak úspěšně překlenout problémy, které oba zmíněné faktory přináší.

Primárním cílem nikdy nebylo skenování všech dostupných URL, ale především hodnotného obsahu. To s sebou přináší aplikaci různých heuristik, které napomáhají eliminovat skenování webového spamu. Webový crawler je tak centrální komponentou komerčních vyhledávacích strojů, a proto se i jeho přesný popis stal obchodním tajemstvím. Mnohé dostupné informace pak neposkytují dostatečnou míru přesnosti, která by dovolila s jejich znalostí postavit totožný ekvivalent.

Existují tak vedle sebe velmi výkonné komerční systémy s omezeným množstvím veřejně známých detailů, a systémy konstruované v akademické sféře pro potřeby výzkumu. Bohužel, výkon akademických produktů mnohdy výrazně zaostává za jejich komerčními protějšky – zejména co do rychlosti procházení webu. Na druhou stranu jsou často dostupné včetně kompletních zdrojových kódů.

Z volně dostupných zdrojů je možné získat informace především o starších crawlerech jako WebCrawler, Google, Mercator [4], Larbin, WebBase crawler [5], Xyro [8], Nutch, ad. Některé z nich budou uvedeny v praktickém výkonovém srovnání v tomto článku.

Kromě samozřejmých nároků na rychlost skenování se jedná i o schopnost přizpůsobení se frekvenci změn na zdrojích informací, včasné reakci na síťové problémy, efektivní správu báze známých zdrojů, eliminaci návštěv duplicitních zdrojů a vyhýbání se rozličným pastem. Bohužel, tyto vlastnosti souvisí s dlouhodobějším provozem jednotlivých strojů, a proto se jimi nebudeme detailněji zabývat – popíšeme pouze některé základní postupy.

1.1 Specifika webového prostředí

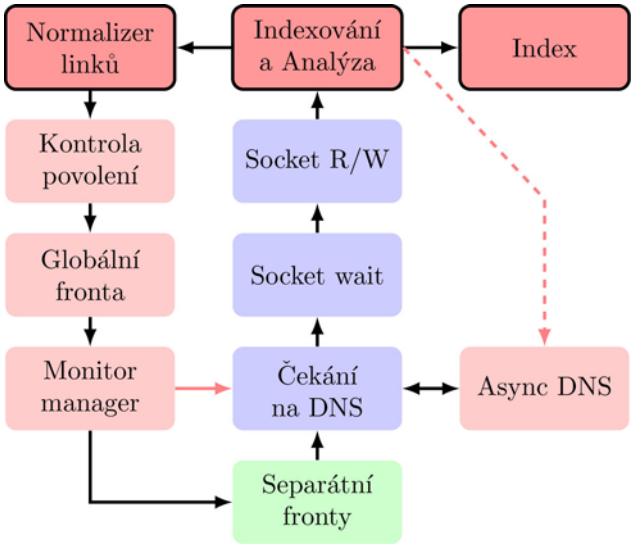
Hlavním problémem, se kterým se webový crawler může setkat, je způsoben nepsaným pravidlem o „přiměřené“ míře obtěžování, kterou může způsobovat svým cílům. Je tak uměle limitován počet spojů na konkrétní IP adresu, respektive (virtuální) webový server. To vede k tomu, že robot stahuje souběžně a krátkodobě z různých zdrojů. Přímým důsledkem je pak snížení efektivity práce keší, které robot používá pro správu URL, plánování atp.

Zároveň je ale nutná i rychlá reakce na případné síťové problémy anebo výpadky zdrojů. Práce keší je pak o to náročnější, neboť robot sám není schopen dopředu predikovat výpadky, které jsou mimo jeho sféru vlivu. Ve výsledku tak křivka popisující průtok dat robotem silně kolísá, což je často příznak ne zcela optimální navigace po webu.

Obecně platí, že je konstrukce webového crawleru snadná. Obtíže nastávají jakmile má být příslušná implementace provozována na skutečném, „nekonečném“ webu. Přispívá k tomu zhoršená efektivita práce s pevnými disky, která je implikována nutností uchovávat datové struktury pojímající velké webové oblasti. Rovněž stabilita a ochrana proti výpadkům způsobuje nárůst náročnosti [9], nehledě na nutnost pravidelné údržby výsledného systému. Doba životnosti bývá v tomto případě odhadována na 12–18 měsíců [6].

2 Architektura, práce crawleru

Nejobecnější strukturu robota představuje obrázek 1. Všechny odkazy na zdroje vstupují do systému skrze normalizátor. Ten je zodpovědný za převod odkazů



Obr. 1: Obecná struktura robota

(URL) do kanonického tvaru. Výsledný tvar nemusí být jen korektní aplikací standardů/doporučení (například eliminace `/./` v URL), ale může být veden i snahou o snížení počtu duplikátů[1]. Mezi ne zcela korektní, ale z praktického pohledu vhodné modifikace, patří [2]:

- odstranění `index.htm` anebo `index.html` s eliminací duplicit až 36 %,
- přidání `www` k URL s doménou stroje druhého řádu s eliminací duplicit až 30 %.

Naproti tomu líbivé modifikace, které za název souboru bez přípony připojí lomítko, vedou jen k 5% eliminaci duplicit. Cena za těchto několik procent je potencionální vytvoření chybných URL, což nemusí být vždy žádoucí.

Normalizovaná URL procházejí kontrolorem, který ověřuje zda je skutečně chceme monitorovat. Pokud ano, tak jsou dočasně skladovány ve velkém zásobníku URL adres. Ten slouží jako keš, protože dále jsou již URL adresy řazeny do front příslušejících konkrétním webovým strojům a zatřiďování každého jednotlivého URL by bylo značně neefektivní.

[1] Stránka s totožným obsahem, která je dostupná pod různými URL.

2.1 Plánovač

Proces vlastního převodu URL z velkého zásobníku do separátních front (SF) nastává, pokud se v nich nenachází dostatečná zásoba URL ke stahování. Spouštění tohoto procesu je v náplni „Monitor manageru“.

Monitor manager je také možné (pro jednoduchost) integrovat s plánovačem. Ve většině současných robotů neprobíhá plánování na jedné úrovni, spíše se používá technika dlouhodobého a krátkodobého plánování. V rámci dlouhodobého plánování dochází k uspořádávání fronty webových serverů, zatímco v u krátkodobého plánování přímo konkrétních stránek na zvoleném webovém serveru. V dlouhodobém výhledu tak plánovač ovlivňuje který webový server bude skenován, krátkodobá strategie pak udává způsob volby konkrétních webových adres pro skenování.

Jednotlivé plánovací strategie byly podrobeny zkoumání v crawleru WIRE [1]. Doporučovaná strategie volí prioritně webové fronty serverů s větším počtem URL, a v těchto separátních frontách pak postupuje strategií do hloubky.

V praktickém nasazení se nám u stroje Egothor2 osvědčilo kombinovat takové plánování s množstvím chybových HTTP nebo síťových stavů. Robot tak dokázal stahovat téměř konstantní rychlostí bez výrazných výkyvů, což vedlo k nárůstu průtoku dat robotem přibližně o 10 %.

2.2 Zpracování front

Každá z front, příslušející konkrétnímu webovému stroji v internetu, začíná své zpracování vyřešením DNS dotazu na odpovídající *hostname*. Robot Mercator, a po něm mnohé další systémy, začaly důsledně využívat asynchronního dotazování DNS spolu s včasnou signalizací objevených DNS jmen. Tím se podařilo zredukovat čas strávený čekáním v DNS modulu až trojnásobně (v případě Mercatoru z 87 % na 25 %).

Jsou-li již k dispozici konkrétní IP adresy, je možné frontu číst a stahovat příslušný obsah webu. V tomto místě se konstrukčně liší dva možné přístupy: buď se využívá výpočetních vláken a blokovaných volání na socketech, anebo se využívají neblokované sockety.

Výpočetní vlákna mají zásadní nevýhodu: dotahují relativně nezávisle a z toho plynou náhodné přístupy do repository, kam se stažený obsah ukládá; zámky nad úložištěm a nároky na správu vláken tak vedou k razantnímu snížení výkonu. Se strojem Egothor jsme například nad Pentiu 733 MHz (Linux, SCSI) dosahovali až o řád horších výkonů v porovnání s neblokovanými sockety.

Stažený obsah postupuje do analyzátoru, který z něj vydestiluje další zdroje (URL). Nově objevená URL pak postupují do normalizátoru a celý cyklus se tím uzavírá. Naproti tomu známá URL jsou pouze oznamována plánovači, který podle četnosti jejich výskytu může modifikovat plánovací strategii.

Posledně jmenovaný krok vyžaduje získání odpovědi na dvě otázky, které chápejme jako jedinou: *Je dané URL v našem systému? Pokud ano, jaké interní označení (číslo) má přiřazeno?*

Ty jsou velice klíčové pro skutečný výkon robota. Představme si, že robot pracuje rychlostí 1 000 cílů za sekundu, a na každém z cílů získá 10 dalších odkazovaných (typický webový příklad). Pak to znamená až 10 000 otázek za sekundu. Protože pracujeme ve velkém objemu, pravděpodobně nevystačíme s RAM. Dále uvažme, že mnoho těchto odkazů bude na cíle blízké výchozímu zdroji, tj. na stejném serveru. Přesto je reálné předpokládat, že alespoň 1 % odkazů takový předpoklad nesplní. Ty pak mohou zapříčinit úplnou saturaci robota, neboť externí datová struktura nebude schopna zmíněný vytrvalý nápor pojmout.

Nejjednodušším řešením je pak využít bitovou mapu v RAM a interní označení stanovovat hešovací funkcí nad normalizovaným tvarem URL. Tento postup zvolil například Larbin. Bohužel, ve velkých objemech bývá problém nalézt vhodnou hešovací funkci s málo kolizemi.

Jedno z řešení je využití externí datové struktury ve spoluprácí s vhodnou keší. V projektu Dominos [3] se osvědčila struktura Judy-array (http://judy.sourceforge.net/).

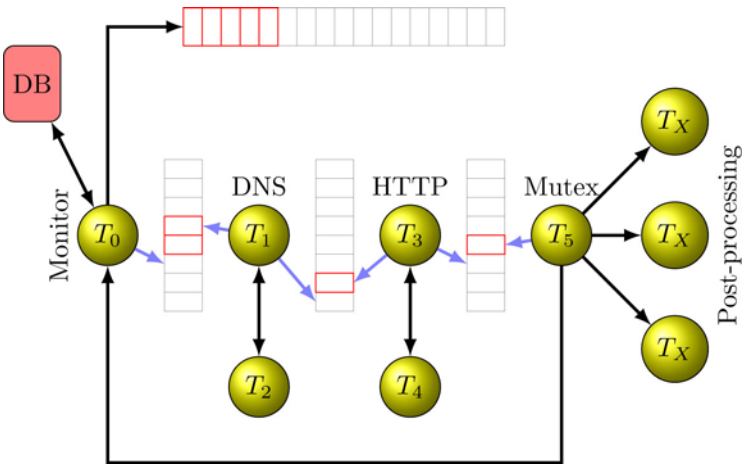
3 Praxe

Seznam věřejně publikovaných výsledků rychlostí webových robotů popisuje tabulka 1. U každého z nich je uvedeno na kolika fyzických strojích a CPU pracoval. Počet str/sec a vlastní tok je vždy přepočítán na 1 CPU. Není-li počet CPU znám, pak je přepočet uskutečněn na počet fyzických strojů crawleru.

Uvedené hodnoty není možné brát absolutně, neboť probíhaly v rozdílných časech a na různých oblastech webu.

Tabulka 1: Známé rychlosti robotů

Robot	#strojů/CPU	str/sec	tok (MB/s)	rok
Googlebot	4/?	25–32	0,2	1998
VN	9/14	2?	0,1	2000
Mercator	4/8	72	1,72	2001
Xyro	4/?	12	?	2001
Nutch	1/?	?	0,5	2004
Become.com	50/?	100	0,3?	2004
Dominos	5/?	154	0,9	2004
Larbin	1/2	80	2,1	2006
Egothor 2	1/2	100	5,0	2007



Obr. 2: EGOTHOR2 – konkrétní realizace robota

3.1 Egothor2

Dále se zaměříme na robota, který byl implementován v rámci projektu Egothor2 na MFF UK. Jeho vnitřní strukturu popisuje obrázek 2.

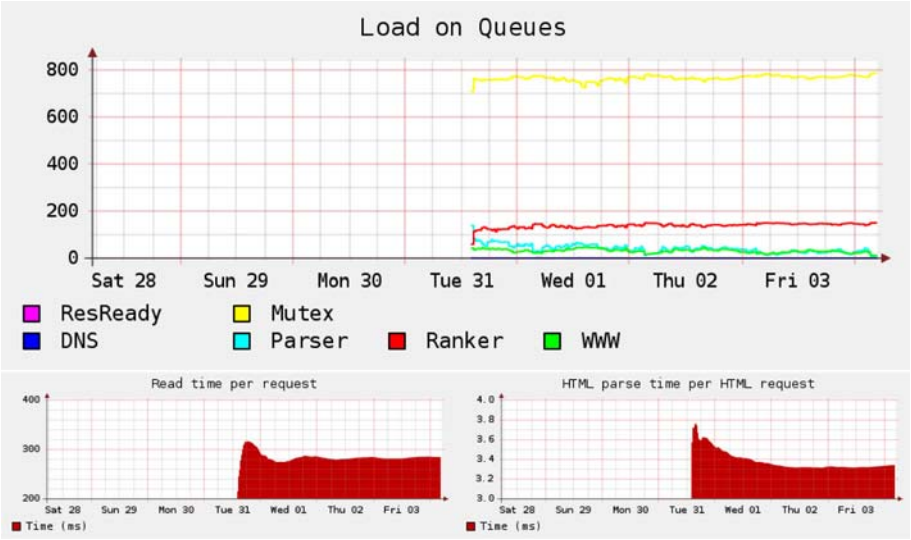
Samotný robot je implementován se šesti základními vlákny, na obrázku označenými jako $T_{0,1,\ldots,5}$. Vlákno T_0 plní funkci „Monitor managera“: z DB separátních front vybírá vhodné kandidáty, které umísťuje do pole „zpracovávaných“ front (PZF). Ty pak postupují k vláknu T_1, které je zodpovědné za dotazování DNS. Po úspěšném získání IP adres(y) postupuje fronta k vláknu T_3, které je zodpovědné za získání cílového dokumentu (určen top-prvkem dané fronty). Dokument dále postupuje ke zpracování do vlákna T_5. Zde dojde k extrakci odkazů, a případně i dodatečnému zpracování vlákny T_X[2]. Příslušná fronta se pak vrací k T_0, které rozhodne zda se bude pokračovat přesunem k T_3, anebo dojde k odložení fronty, tj. vyhození z PZF.

Kvůli sledování prodlev jsou vlákna T_1 a T_3 vybavena dalším pomocným vláknem (T_2 a T_4). Jeho činnost je však zcela minimální.

Věškeré předávání front mezi vlákny probíhá skrze cyklické buffery. Pokud by docházelo k jejich saturaci, tak systém může rozhodnout o automatickém naklonování vlákna, které nestíhá odebírat prvky.

Reálný provoz na jednotlivých frontách popisuje obrázek 3. Jednalo se o test s omezením vstupního toku na 4 MB/s nad doménou CZ.

[2]Tyto vlákna jsou využívána vyhledávacím strojem pro značkování „zajímavých“ dokumentů, stanovování jazyka atp.



Obr. 3: EGOTHOR2 – grafy provozu

Je patrné, že největší zátěž přichází do vlákna T_5. Toto vlákno není nijak výrazně brzděno parsováním HTTP/HTML, protože rychlost parsování je dle obrázku 3 teoreticky 280 dokumentů za sekundu, zatímco provoz odpovídá toku 80 dokumentů za sekundu.

Příčiny tohoto rozdílu jsou souběžně pracující vlákna, a také dávkové rozřazování URL z velkého zásobníku do separátních front. Bohužel, posledně jmenované činnosti je těžké se vyhnout. Na druhou stranu, systém má po delší době provozu takovou zásobu URL, že již není nutné provádět rozřazování příliš často.

Přiměřená doba rozřazování je ale další otázkou, kterou je zapotřebí vyřešit. Plánování se děje pouze nad již rozřazenými URL, takže nadměrné zvýšení intervalu přesunu URL do separátních front může zpozdit návštěvu nově objevených cílů.

4 Závěr

V rámci tohoto článku byla popsána základní architektura robota, včetně vymezení kritických míst. Byly představeny některé možné varianty řešení a jejich faktický dopad na implementaci robota v projektu Egothor2.

Literatura

[1] Castillo, C. *Effective web crawling.* SIGIR Forum, 39(1), 55–56, ACM Press, NY, USA, 2005.

[2] Gomes, D., Silva, M. J. *The Viuva Negra crawler.* FI-FC UL, TR-2006-06-21, Lisboa, 2006.

[3] Hafri, Y., Djeraba, C. *Dominos: A New Web Crawler's Design.* IWAW, Bath, UK, 2004.

[4] Heydon, A., Najork, M. *Mercator: A scalable, extensible web crawler.* WWW 2(4), 219–229, 1999.

[5] Hirai, J., et al. WebBase: A repository of web pages. In *Proc. of the 9th WWW Conf.*, Amsterodam : 2000.

[6] Kahle, B. The internet archive. In *RLG Diginews*, 6(3), 2002.

[7] Lyman, P., Varian, H. R. *How much information*, Berkeley, 2003. http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/

[8] Mignet, L., et al Xyro: The Xyleme Robot Architecture. In *DIWeb*, 91–99, 2001.

[9] Shkapenyuk, V., Suel, T. *Design and implementation of a high-performance distributed web crawler.* Proc. of the 18th Intl. Conf. on Data Engineering, 357–368, San Jose : 2002.

Analýza dat z otevřených zdrojů

Iveta Mrázová

E-mail: iveta.mrazova@mff.cuni.cz

Abstrakt

Podstata úspěšné analýzy dat z otevřených zdrojů spočívá v efektivním zpracování velkého množství údajů, a to pokud možno bez výraznějších zásahů uživatele. Cílem takové analýzy je najít vzájemné vztahy mezi zpracovávanými daty a tuto informaci smysluplně interpretovat. V příspěvku představíme některé ze základních metod použitelných pro analýzu dat z otevřených zdrojů – mezi jinými klasifikační techniky založené na principu Bayesovských klasifikátorů a vrstevnaté neuronové sítě typu zpětného šíření. Samoorganizační techniky zahrnují Kohonenovy mapy a asociační pravidla.

1 Úvod

Podstata úspěšné analýzy dat z otevřených zdrojů spočívá v efektivním zpracování velkého množství údajů, a to pokud možno bez výraznějších zásahů uživatele. Cílem takové analýzy je najít vzájemné vztahy mezi zpracovávanými daty a tuto informaci smysluplně interpretovat. Výsledky získané během analýzy ovšem může být poměrně obtížné vhodně vizualizovat. Významným požadavkem bývá také snadná použitelnost vyvinutých metod i pro pozměněné požadavky, např. nová data.

Data běžně dostupná z otevřených zdrojů mají typicky různé vlastnosti a různý formát. Velká část informací je uložena v textové podobě. S rozvojem výpočetní kapacity běžně dostupných informačních technologií však dochází i k masivnímu rozšiřování netextových informací, např. sdílení obrázků, akustických záznamů, videí apod. K úlohám řešeným v rámci analýzy dat z otevřených zdrojů tedy patří zejména klasifikace, shlukování, analýza (sociálních) vazeb a vyhledávání (např. v textech).

Pro automatické zpracování dat z otevřených zdrojů se vzhledem k výše uvedeným důvodům jeví jako nejvhodnější techniky založené na principu strojového

Příspěvek vznikl s podporou grantu č. 358/2006/A INF/MFF GA UK.

učení [4]. Při tzv. učení s učitelem v podstatě hledáme výpočetní model určující hodnotu cílového atributu na základě ostatních vstupních atributů. K učení modelu se používá tzv. trénovací množina vzorů, pro kterou známe hodnoty cílových atributů. Metody tzv. učení bez učitele jsou určené pro případy, kdy cílem není odhad hodnoty cílového atributu, ale analýza dalších vlastností předložené trénovací množiny vzorů – např. rozdělení předkládaných vzorů do shluků (navzájem disjunktních podmnožin vzorů, v nichž jsou si vzory navzájem podobné.

Algoritmy vhodné pro automatickou analýzu dat z otevřených zdrojů obvykle patří k výpočetně náročnějším a volba odpovídající techniky závisí mj. i na charakteru řešené úlohy a použitých dat, která navíc mohou být zatížena vysokou mírou šumu. V příspěvku představíme některé ze základních metod použitelných pro analýzu dat z otevřených zdrojů – mezi jinými klasifikační techniky založené na principu Bayesovských klasifikátorů a vrstevnaté neuronové sítě typu zpětného šíření. Samoorganizační techniky zahrnují Kohonenovy mapy a asociační pravidla.

2 Bayesovské klasifikátory

Idea Bayesovských klasifikátorů je založena na základních principech teorie pravděpodobnosti [2]. Charakteristiky zpracovávaných dat jsou včetně cílového atributu chápány jako náhodné veličiny. Model Bayesovského klasifikátoru pak vychází z Bayesovy věty pro výpočet aposteriorní pravděpodobnosti $\mathrm{P}(H|E)$ na základě znalosti apriorní pravděpodobnosti $\mathrm{P}(H)$ a podmíněné pravděpodobnosti $\mathrm{P}(E|H)$, která určuje pravděpodobnost pozorování jevu E za předpokladu, že platí hypotéza H:

$$\mathrm{P}(H|E) = \frac{\mathrm{P}(E|H) \cdot \mathrm{P}(H)}{P(E)}$$

Hodnota cílového atributu pak bude odpovídat indexu hypotézy s maximální aposteriorní pravděpodobností:

$$H_{\max} = \underset{H_j}{\operatorname{argmax}}\ \mathrm{P}(H_j)\mathrm{P}(E|H_j).$$

V tomto vztahu označuje H_j možné hypotézy (resp. možné hodnoty cílového atributu), E odpovídá hodnotám (resp. vektoru hodnot) ostatních atributů.

K nejjednodušším a pravděpodobně také nejčastěji používaným Bayesovským modelům patří tzv. naivní Bayesovský klasifikátor. Jeho základní myšlenka vychází z předpokladu, že všechna pozorování, na kterých je klasifikace založena jsou navzájem podmíněně nezávislá [8]. To umožňuje zjednodušit výpočet aposteriorních pravděpodobností. Výraz $\mathrm{P}(E_1, E_2, \ldots, E_K|H)$ lze totiž díky předpokladu podmíněné nezávislosti pozorování zapsat jako $\prod_{k=1}^{K} \mathrm{P}(E_k|H)$. Vzorec pro

výpočet největší aposteriorní pravděpodobnosti se pak zjednoduší na:

$$H_{\max} = \underset{H_j}{\operatorname{argmax}} \mathrm{P}(H_j) \prod_{k=1}^{K} \mathrm{P}(E_k|H_j).$$

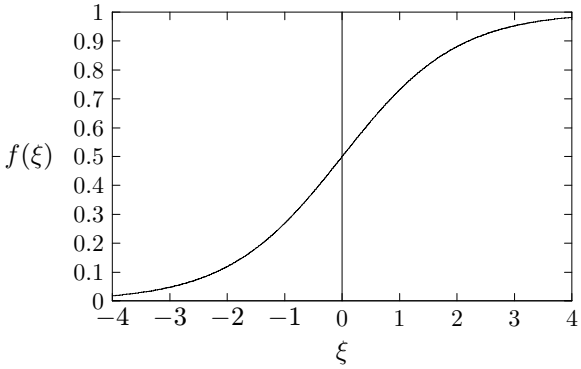
V praxi je předpoklad podmíněné nezávislosti pozorování splněn jen velmi zřídka – a z tohoto důvodu je klasifikátor označovaný jako „naivní“. Naivní Bayesovský klasifikátor ale (při vhodné volbě sady pozorování) dává překvapivě dobré výsledky i v případě, že tento předpoklad splněn není. Navíc jde o velmi rychlou a poměrně přesnou metodu, která je schopna správně klasifikovat i vícerozměrné vzory a nevyžaduje přitom příliš velkou trénovací množinu dat. Předpoklad podmíněné nezávislosti pozorování totiž snižuje nároky na množství dat potřebných v trénovací množině pro naučení klasifikátoru [4]. Obtížnější je na druhou stranu použití pravděpodobnostních modelů pro úlohy, kde je obor hodnot některého z atributů nekonečný.

3 Vrstevnaté neuronové sítě

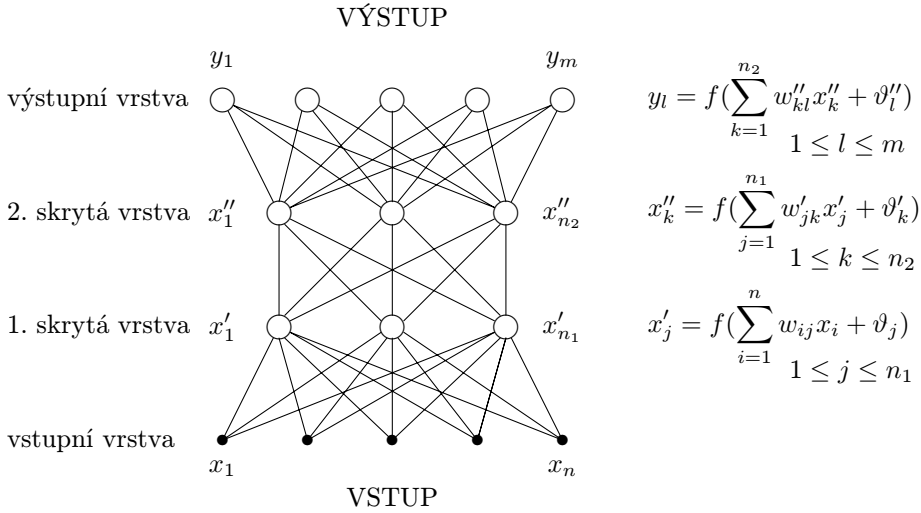
V oblasti umělých neuronových sítí patří v současné době k nejpoužívanějším model vrstevnatých neuronových sítí typu zpětného šíření. Někdy se tento model označuje také jako BP-sítě. Oblast využití tohoto modelu sahá od počítačového vidění přes robotiku, řízení, medicínu a ekonomii až po umělou inteligenci. Obvykle se tyto sítě používají ke klasifikaci vzorů s pevným počtem příznaků, které mohou nabývat i spojitých hodnot. Vrstevnaté neuronové sítě typu zpětného šíření se skládají ze vstupní vrstvy (obsahující všechny vstupní neurony), několika mezilehlých vrstev (tzv. skrytých vrstev obsahujících skryté neurony) a výstupní vrstvy (obsahující všechny výstupní neurony). Umělé neuronové sítě tohoto typu však neobsahují synaptické váhy spojující neurony uvnitř jednotlivých vrstev ani synaptické váhy jdoucí z vyšších vrstev do nižších nebo přesahující jednu anebo více vrstev. Aktivita neuronů se v každé vrstvě určí podle:

$$\xi_j = \sum_i y_i w_{ij} \qquad \text{a} \qquad y_j = f(\xi_j) = \frac{1}{1 + \mathrm{e}^{-\lambda \xi_j}}$$

Celkový vážený vstup ξ_j neuronu j (tzv. potenciál neuronu) je dán součtem výstupních aktivit y_i neuronů předchozí vrstvy, které jsou vynásobeny váhou příslušné synapse w_{ij} mezi neuronem i a j. Výstupní aktivita neuronu j, y_j, je určena nelineární (např. sigmoidální – viz obázek 1) přenosovou funkcí f a potenciálem daného neuronu. Neurony uvnitř jedné vrstvy přitom mění svou aktivitu paralelně, a to postupně po jednotlivých vrstvách od vstupní vrstvy směrem k výstupní. Parametr λ se nazývá strmost přenosové funkce (a může mít případně různou hodnotu pro každý neuron).



Obr. 1: **Sigmoida:** Výstupní hodnotu neuronu s potenciálem ξ lze určit pomocí sigmoidální přenosové funkce $f(\xi)$: $y = f(\xi) = 1/(1 + \mathrm{e}^{-\xi})$.



Obr. 2: **Třívrstvá síť** s n vstupními neurony, m výstupními neurony a dvěma skrytými vrstvami s n_1, resp. n_2 skrytými neurony. Jako nelinearitu lze použít například sigmoidální přenosovou funkci znázorněnou na obrázku 1. Hodnoty x'_j a x''_k zde představují výstupy neuronů z první, resp. druhé skryté vrstvy, ϑ'_k a ϑ''_l značí prahy těchto neuronů, w_{ij} označuje váhu synapse mezi vstupním neuronem a neuronem z první skryté vrstvy a w'_{ij}, resp. w''_{ij} jsou vahami mezi první a druhou skrytou vrstvou, resp. mezi druhou skrytou vrstvou a vrstvou výstupní.

Pro vrstevnatou neuronovou síť s n vstupními a m výstupními neurony představuje vstupní vzor vstupní vektor $\vec{x} \in R^n$ zpracovávaný sítí. Požadovaný výstup $\vec{d} = (d_1, \ldots, d_m)$ tvoří pořadované výstupy neuronů výstupní vrstvy a pro daný vstupní vzor představuje skutečný výstup sítě vektor $\vec{y} = (y_1, \ldots, y_m)$ tvořený skutečnými výstupy neuronů výstupní vrstvy. Při učení je síti předkládán vždy vstupní vzor zároveň s požadovaným výstupem. V procesu učení se vrstevnaté neuronové sítě „učí správně reagovat na možné vstupy“, a to postupným předkládáním vzorů z trénovací množiny (učení s učitelem). Přitom se pomocí algoritmu zpětného šíření adaptují váhy spojů mezi jednotlivými neurony sítě.

Cílem algoritmu zpětného šíření [11] je nalézt takovou množinu vah, která by zaručovala pro všechny vstupní vzory z trénovací množiny to, že skutečný výstup dané neuronové sítě bude stejný jako její požadovaný výstup. Úloha přitom nespecifikuje ani skutečnou ani požadovanou aktivitu skrytých neuronů. Teprve v procesu učení se musí síť „sama rozhodnout“, za jakých okolností má být ten který skrytý neuron aktivní a přispět tak k dosažení požadovaného chování sítě. Požadované chování sítě lze formulovat pomocí tzv. cílové funkce, jejíž hodnota by se měla v procesu učení zmenšovat. Pro konečnou množinu trénovacích vzorů ve tvaru (vstupní_vzor/požadovaný_výstup) lze cílovou funkci sítě vyjádřit pomocí rozdílu mezi skutečným a požadovaným výstupem u každého předloženého vzoru. Cílová funkce E, označovaná někdy i jako chybová anebo účelová funkce, je pak definovaná jako:

$$E = \frac{1}{2} \sum_p \sum_j (y_{j,p} - d_{j,p})^2$$

kde p je index označující předkládaný vzory z trénovací množiny, j indexuje výstupní neurony, y představuje skutečný výstup příslušného neuronu a d jeho požadovaný výstup.

K minimalizaci chybové funkce E se používá gradientní metoda. Přitom je nutné nastavit v průběhu učení hodnoty všech synaptických vah v síti tak, aby byla hodnota funkce E co možná nejmenší. Hodnotu každé váhy w_{ij} je tedy třeba změnit o hodnotu odpovídající záporné parciální derivaci E podle této váhy, $-\partial E/\partial w_{ij}$. Označíme-li tento člen jako $\Delta_E w_{ij}$, platí:

$$\Delta_E w_{ij} = -\frac{\partial E}{\partial w_{ij}}$$

Prahy neuronů přitom lze reprezentovat pomocí vah vycházejících z fiktivního neuronu s konstantním výstupem rovným 1. Pro jednoduchost dále nebudeme uvažovat index p ani u požadovaných, ani u skutečných výstupních hodnot neuronů, d and y. Z podobných důvodů budeme dále označovat neurony z vrstvy pod příslušným neuronem j jako i a neurony z vrstvy nad neuronem j jako k. K výpočtu hodnot $\Delta_E w_{ij}$ lze použít standardní algoritmus zpětného šíření. Pro

předložený trénovací vzor je nejprve třeba určit skutečný výstup sítě a ten potom porovnat s požadovaným výstupem. Na základě zjištěné odchylky pak lze pro každou váhu v dané síti spočítat zápornou parciální derivaci E podle této váhy, $\Delta_E w_{ij}$:
– pro výstupní vrstvu

$$\begin{aligned}\Delta_E w_{ij} &= -\frac{\partial E}{\partial w_{ij}} = -\frac{\partial E}{\partial \xi_j}\frac{\partial \xi_j}{\partial w_{ij}} = -\frac{\partial E}{\partial \xi_j}\frac{\partial}{\partial w_{ij}}\sum_{i'} w_{i'j}y_{i'} = \\ &= -\frac{\partial E}{\partial \xi_j}y_i = -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \xi_j}y_i = (d_j - y_j)\lambda y_j(1-y_j)y_i = \\ &= \delta_j y_i\end{aligned}$$

– a pro skryté vrstvy

$$\begin{aligned}\Delta_E w_{ij} &= -\sum_k \frac{\partial E}{\partial \xi_k}\frac{\partial \xi_k}{\partial y_j}\frac{\partial y_j}{\partial \xi_j}y_i = -\left(\sum_k \frac{\partial E}{\partial \xi_k}\frac{\partial}{\partial y_j}\sum_j w_{jk}y_j\right)\frac{\partial y_j}{\partial \xi_j}y_i = \\ &= -\left(\sum_k \frac{\partial E}{\partial \xi_k}w_{jk}\right)\frac{\partial y_j}{\partial \xi_j}y_i = \left(\sum_k \delta_k w_{jk}\right)\lambda y_j(1-y_j)y_i = \\ &= \delta_j y_i\end{aligned}$$

Výraz pro aktualizaci vah tedy bude mít tvar:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha\Delta_E w_{ij} = w_{ij}(t) + \alpha\delta_j y_i$$

V tomto vztahu odpovídá člen δ_j:

$$\delta_j = \begin{cases} (d_j - y_j)y_j(1-y_j)\lambda & \text{pro výstupní neuron} \\ \lambda y_j(1-y_j)\sum_k \delta_k w_{jk} & \text{pro skrytý neuron} \end{cases}$$

w představuje jednotlivé váhy, resp. prahy sítě, i a i' označuje neurony propojené s neuronem j vahou w_{ij}, resp. $w_{i'j}$. k indexuje neurony ve vrstvě nad neuronem j. d_j je požadovaný a y_j skutečný výstup neuronu j. $\xi_j = \sum_{i'} w_{i'j}y_{i'}$ pak označuje potenciál tohoto neuronu. $t+1$ a t představují následující a aktuální cyklus učení. α je konstanta reprezentující parametr učení. Z praktických důvodů je vhodné volit hodnotu α co možná největší (na druhou stranu ovšem i s ohledem na případné oscilace v procesu učení). To obvykle umožňuje rychlejší učení, ačkoliv pro praktické účely je standardní algoritmus zpětného šíření přece jen poněkud pomalý.

Proces učení lze urychlit např. pomocí tzv. momentu. Tato metoda zpravidla vede i k omezení oscilací během učení a vyžaduje pouze drobnější modifikaci adaptačního pravidla, které nyní bere v úvahu i změny vah z předchozího cyklu:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha\delta_j y_i + \alpha_m(w_{ij}(t) - w_{ij}(t-1))$$

V této rovnici představuje w odpovídající váhy, resp. prahy. i a k indexují neurony spojené s neuronem j pomocí váhy w_{ij}, resp. w_{jk}. y_j je skutečný výstup neuronu j a δ_j odpovídá příslušnému chybovému členu. $t+1$, t a $t-1$ indexují následující, aktuální a předchozí váhy. α označuje parametr učení a α_m je konstanta mezi 0 a 1, která určuje velikost vlivu předchozí změny vah při jejich současné aktualizaci. Obvykle se tento člen označuje jako moment. Vliv momentu se projeví především v oblastech pomalé konvergence algoritmu s téměř konstantní hodnotou gradientu, kde vede ke zvětšování prováděných změn. Naproti tomu v oblastech s prudkými změnami hodnot gradientu $(+/-)$ vede použití této modifikace standardního algoritmu zpětného šíření k omezení oscilací v procesu učení.

Algoritmus zpětného šíření

Algoritmus zpětného šíření je v podstatě iterativní gradientní metoda navržená k minimalizaci střední kvadratické odchylky mezi skutečným a požadovaným výstupoem BP-sítě. Jeho použití tedy předpokládá hladkou nelineární přenosovou funkci – pro naše účely použijeme sigmoidální přenosovou funkci $f(\xi)$ se strmostí 1. Graf této funkce je znázorněn na obrázku 1:

$$f(\xi) = \frac{1}{1+\mathrm{e}^{-\xi}}, \text{kde} \qquad \xi = \sum_k w_k x_k$$

Krok 1: Inicializace vah a prahů
Zvol hodnoty vah a prahů v séti jako malé náhodné hodnoty.

Krok 2: Předlož nový trénovací vzor
Předlož síti vstupní vzor ve tvaru $x_1, x_2, \ldots, x_n$ a specifikuj požadované výstupy $d_1, d_2, \ldots, d_m$.

Krok 3: Spočítej skutečné výstupy
Skutečné výstupy $y_1, y_2, \ldots, y_m$ se určí pomocí sigmoidální přenosové funkce a vzorců uvedených v obrázku 2. Dále spočítej hodnotu chybové funkce E a **Skonči**, jestliže je tato hodnota dostatečně malá anebo byl proveden dostatečný počet cyklů.

Krok 4: Aktualizace vah a prahů

Při aktualiyaci synaptických vah postupuj od výstupní vrstvy směrem ke vstupní. Váhy se adaptují podle:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha\delta_j y_i$$

V tomto vztahu představuje $w_{ij}(t)$ váhu ze skrytého neuronu anebo vstupního neuronu i v čase t, y_i označuje skutečný výstup neuronu i, α je parametr učení a δ_j je chybový člen odpovídající neuronu j. Pokud leží neuron j ve výstupní vrstvě,

$$\delta_j = y_j(1-y_j)(d_j - y_j),$$

kde d_j je požadovaný výstup neuronu j a y_j je jeho skutečný výstup.

Jestliže leží neuron j ve skryté vrstvě,

$$\delta_j = y_j(1-y_j)\sum_k \delta_k w_{jk},$$

kde k je index pro neurony z vrstvy nad neuronem j. Prahy neuronů se adaptují podobným způsobem, vezmeme-li v úvahu, že se jedná v podstatě o váhy s konstantními vstupy. Konvergence bývá ovšem často rychlejší, použijeme-li modifikované adaptační pravidlo s momentem:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha\delta_j y_i + \alpha_m(w_{ij}(t) - w_{ij}(t-1)),$$

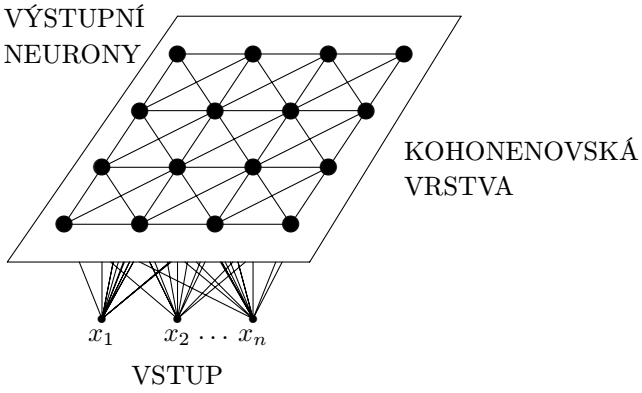
přičemž $0 < \alpha_m < 1$.

Krok 5: Přejdi ke Kroku 2 a opakuj cyklus

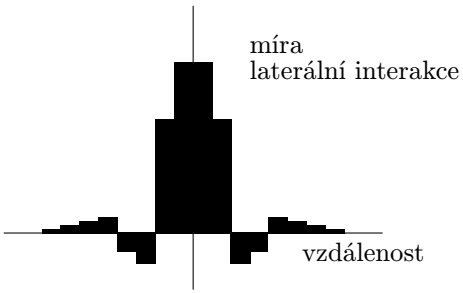
Standardní algoritmus zpětného šíření má mnoho předností. Jeho aproximační schopnosti a schopnost vytvořit (alespoň v některých případech) vhodnou interní reprezentaci znalostí patří k nejdůležitějším vlastnostem vrstevnatých neuronových sítí typu zpětného šíření a přispívají k jejich dobrým generalizačním schopnostem. Obvykle je však velice obtížné odhadnout význam jednotlivých skrytých neuronů. Vhodnou strategií učení však lze získat sítě s transparentní strukturou a interní reprezentací znalostí [10].

K velkým nedostatkům této metody naopak patří výskyt tzv. lokálních minim a pomalá konvergence obzvláště při řešení náročných úloh pomocí sítí s téměř optimální architekturou [12]. Obecně přitom nelze konvergenci algoritmu zaručit.

Kritéria pro hodnocení vlastností a funkce vrstevnatých neuronových sítí dále zahrnují robustnost sítě vzhledem k malým odchylkám předkládaných vstupních vzorů a možnost opětovného využití již natrénovaných sítí za změněných podmínek. Změněné podmínky tu často představuje modifikovaná trénovací množina a/nebo pozměněné požadavky na funkci sítě.



Obr. 3: **Kohonenova mapa** s výstupními neurony uspořádanými do dvourozměrné mřížky. Každý vstupní neuron je přitom propojen se všemi výstupními neurony



Obr. 4: **Funkce laterální interakce** tvaru „mexického klobouku“

4 Samoorganizace a Kohonenovy mapy

Při myšlení i podvědomém zpracování informací je patrná obecnáá tendence komprimovat zpracovávané informace vytvářením redukovaných reprezentací

nejrelevantnějších faktů pokud možno beze ztráty znalosti jejich vzájemných (topologických) vztahů. T. Kohonen navrhl v [5] strategii pro vytváření tzv. samoorganizujících se příznakových map podobných těm, které fungují v mozku. Samoorganizující se příznakováámapa rozprostře neurony ve vstupním prostoru tak, že jsou soustředěny v oblastech s nejvyšší hustotou vstupních vzorů. Přitom by mělo zůstat zachováno vzájemné prostorové uspořádání vstupních vzorů.

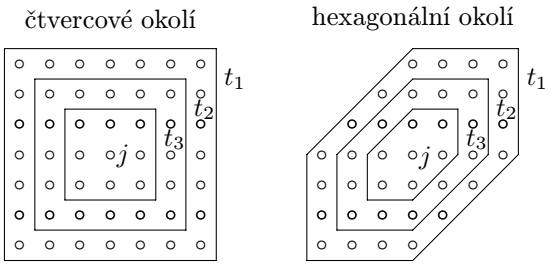
Kohonenův algoritmus adaptuje váhy z obecných vstupních neuronů do výstupních neuronů uspořádaných např. ve dvourozměrné mřížce. Vstupní vrstva je plně propojena s výstupní, tzv. Kohonenovskou vrstvou. Neurony v Kohonenovské vrstvě jsou propojeny s ostatními neurony téže vrstvy v „okolích". Definice sousedů každého neuronu tak určuje topologii Kohonenovy vrstvy. Jsou-li neurony uspořádané obecně do n-rozměrné mřížky a je-li neuron s maximální odezvou na daný vstupní vzor jeho obrazem, označujeme toto zobrazení za uspořádané, jestliže jsou topologické vztahy obrazů a vzorů podobné. V procesu učení jsou síti postupně předkládány jednotlivé vstupní vzory, a to bez specifikace požadovaného výstupu – jedná se tedy o učení bez učitele. Poté, co bylo síti předloženo dostatečné množství vstupních vzorů, by se měly váhy uspořádat tak, že topologicky blízké neurony budou citlivé na „podobné" vstupní vzory.

Při rozpoznávání dostáváákaždý neuron vstupní vrstvy odpovídající vstupní hodnotu $x_i; 1 \leq i \leq n$ a $x_1, x_2, \ldots, x_n \in R$ (tzn., že je přenosová funkce zanedbána). Neuron j v Kohonenově vrstvě z těchto vstupních hodnot spočítá svůj potenciál: $\xi_j = \sum_{i=1}^{n} w_{ij} x_i$; kde $w_{ij} \in R$ jsou váhy synapsí jdoucích ze vstupního neuronu i k výstupnímu neuronu j. Cílem procesu učení je přitom stav, kdy synaptické váhy zkonvergovaly k takovým hodnotám, že je každý neuron citlivý na signály z určité oblasti vstupního prostoru. Maximální odezva neuronu pak odpovídá největší podobnosti mezi vektory $\vec{x} = [x_1, \ldots, x_n]^T$ a $\vec{w_i} = [w_{i1}, \ldots, w_{in}]^T$. Kritériem podobnosti může být např. i Euklidovskáá vzdálenost mezi těmito dvěma vektory. V takovém případě lze neuron c vykazující největší podobnost určit pomocí:

$$\|\vec{x} - \vec{w_c}\| = \min_j \|\vec{x} - \vec{w_j}\|.$$

Neuron s maximální odezvou tak reprezentuje střed oblasti vykazující vysokou míru vzájemné podobnosti – tzv. „okolí". Mezi neurony v takovém „okolí" pak mohou existovat následující typy laterální interakce (tzn. stupně vzájemného ovlivnění):

- oblast laterální excitace s malým rozsahem
- excitaźční oblast je obklopena oblastí inhibiční akce
- oblast inhibiční akce obklopuje oblast menší excitační akce



Obr. 5: **Příklady topologického okolí** ($t_1 < t_2 < t_3$). Okolí je na počátku velké a v čase se postupně zmenšuje

Stupeň laterální interakce se obvykle popisuje pomocí funkce tvaru „mexického klobouku“.

Obecně je δ-okolím prvku p z množiny X množina všech takových prvků množiny X, které mají od prvku p vzdálenost menší než δ. V případě Kohonenových map je vhodné definovat topologické okolí neuronu c jako funkci indexu neuronu v diskrétním čase t, $N_c = N_c(t)$. Dobrých výsledků lze dosáhnout při velkém okolí na zaáčtku adaptace, které se postupě zmenšuje. Adaptační pravidlo pro Kohonenovy mapy lze přitom formulovat ve tvaru:

$$\begin{aligned} \Delta w_{ik}(t) &= \alpha(t)(x_i(t) - w_{ik}(t)) \quad \text{pro} \quad k \in N_c \\ \Delta w_{ik}(t) &= 0 \qquad\qquad\qquad\qquad\quad \text{pro} \quad k \notin N_c, \end{aligned}$$

kde $\alpha(t)$ označuje hodnotu parametru učení v kroku t a posloupost $\{\alpha(t); t = 0, 1, \ldots; 0 < \alpha(t) < 1\}$ je obvykle funkce pomalu klesající v čase. Koeficienty $\alpha(t)$, které se někdy označují i jako koeficienty bdělosti, by navíc měly splňovat následující podmínky [5]:

$$\sum_{s=0}^{\infty} \alpha(s) = \infty \quad \text{a} \quad \sum_{s=0}^{\infty} \alpha(s)^2 < \infty;$$

Proces adaptace se automaticky zastaví při $\alpha(t) = 0$. Obecně pak mají váhové vektory $\vec{w}_i$ tendenci uspořádat se podle své vzájemné podobnosti a aproximovat tak hustotu rozložení vzorů v příznakovém prostoru.

V procesu samoorganizace přitom probíhají dvě protikladné tendence. Množina vah má jednak tendenci popsat hustotu vstupních vzorů v příznakovém prostoru. Na druhou stranu má však laterální interakce mezi jednotlivými neurony tendenci zachovat kontinuitu v posloupnosti váhových vektorů. Výsledkem je, že rozdělení váhových vektorů bude aproximovat tvar podobný ploše nad příznakovým prostorem, ale bude zároveň hledat optimální orientaci a tvar v příznakovém

prostoru, který nejlépe odpovídá struktuře vstupních vektorů. Rozdělení váhových vektorů pak tedy může pomoci detekovat takové příznaky, ve kterých mají vstupní vzory vysoký rozptyl a které by proto měly být v mapě popsány.

Základní model Kohonenových map lze dále modifikovat [12] – např. lze modelovat efekty „únavy“ pomocí postupného zeslabování vah spojů při vytrvalé aktivitě příslušného neuronu. Shluky neuronů se mohou po nějaké době také rozpadnout a síť může akceptovat nový typ excitace. Jednotlivé modifikace tohoto základního modelu by pak měly brát v úvahu i různé typy mezních efektů. Z aplikačního hlediska jsou Kohonenovy samoorganizující se příznakové mapy vhodné zejména pro předzpracování dat. Při analýze velkého množství dat pak mohou pomoci při výběru signifikantních příznaků zpracovávaných vstupních vzorů metody popisované např. v [7, 9].

Algoritmus učení pro Kohonenovy Mapy

Krok 1: Inicializace vah sítě
Zvol hodnoty vah mezi n vstupními a m výstupními neurony jako malé náhodné hodnoty. Nastav počáteční poloměr okolí.

Krok 2: Předlož nový trénovací vzor
Předlož síti vstupní vzor ve tvaru $(x_1, x_2, \ldots, x_n)$.

Krok 3: Spočítej vzdálenost ke všem neuronům
Pro všechny výstupní neurony spočítej vzdálenost mezi vstupním vzorem a jejich váhovým vektorem $\vec{w}_j$:

$$e_j = \left(\sum_{i=1}^{n} (x_i(t) - w_{ij}(t))^2 \right)^{1/2},$$

kde $x_i(T)$ označuje vstup neuronu i v čase t a $w_{ij}(t)$ odpovídá synaptické váze mezi vstupním neuronem i a výstupním neuronem j v čase t. Za určitých okolností je možné výpočet vzdálenosti modifikovat pomocí váhových koeficientů.

Krok 4: Vyber „vítězný neuron“
Vyber (např. pomocí laterální inhibice) výstupní neuron c s nejmenší vzdáleností e_j od předloženého vstupního vzoru a označ ho jako „vítězný“:

$$e_c = \min_j e_j = \min_j \left(\sum_{i=1}^{n} (x_i(T) - w_{ij}(T))^2 \right)^{1/2}.$$

Krok 5: Aktualizace vah
Uprav váhy vítězného neuronu c a všech neuronů k z jeho okolí, $k \in N_c$. Nové váhy urči podle:

$$w_{ik}(T+1) = w_{ik}(T) + \alpha(T)(x_i(T) - w_{ik}(T)).$$

Koeficient $\alpha(T)$ odpovídá parametru učení ($0 < \alpha(T) < 1$) a klesá v čase. Během učení upravuje vítězný neuron svůj váhový vektor směrem k předloženému vstupnímu vzoru. Totéž platí i pro neurony z okolí vítězného neuronu. Hodnota parametru učení klesá s rostoucí vzdáleností od centra okolí N_c.

Krok 6: Přejdi ke Kroku 2 a opakuj cyklus

5 Asociační pravidla a algoritmus Apriori

K technikám založeným na principu učení bez učitele lze zařadit i asociační pravidla, která se používají pro vyhledávání důležitých a potenciálně zajímavých vztahů mezi jednotlivými atributy ve velkých souborech dat. Asociační pravidla byla původně navržena pro tzv. analýzu nákupního košíku. Výsledky této analýzy pomáhaly prodejcům uspořádat prodávané zboží v regálech, navrhnout vhodnou strategii prodeje a případně i speciální slevy [3]. Extrahovaná pravidla typu $A \Longrightarrow B$ lze interpretovat tak, že pokud si zákazník koupí zboží A, koupí si s velkou pravděpodobností i zboží B. Potom může být výhodné snížit cenu zboží A a naopak zvýšit cenu zboží B. Prodejce může zboží v regálech uspořádat tak, aby položky A i B byly blízko sebe a zákazníkům se usnadnilo nakupování. Alternativou naopak může být také umístit zboží A a B daleko od sebe, aby byl zákazník donucen prohlédnout si i několik dalších regálů se zbožím.

Při analýze nákupního košíku tedy jednotlivé položky obsažené na levé či pravé straně generovaných pravidel odpovídají příslušným produktům nebo nabízeným službám. Trénovací vzory z databáze odpovídají jednotlivým transakcím (nákupům) a každá transakce obsahuje jednu nebo více položek. Typicky jsou všechny atributy binární a jejich hodnota odpovídá přítomnosti (1), resp. nepřítomnosti (0) dané položky v nákupním košíku. Tuto reprezentaci lze, nicméně, zobecnit tak, aby hodnoty jednotlivých atributů reflektovaly i množství nakoupeného zboží. Protože je počet uvažovaných položek typicky velký, zatímco jednotlivé nákupy zřídkakdy překračují několik málo desítek položek, budou vytvářené datové matice nutně řídké. Obrovské množství možných asociačních pravidel, jejichž počet navíc exponenciálně narůstá s počtem uvažovaných položek, tedy bude nutné smysluplně zredukovat [6].

V obecném případě nás totiž budou zajímat pouze taková asociační pravidla, která se budou týkat dostatečného počtu trénovacích vzorů (např. více než 1 %) a budou dostatečně přesná (např. více než 80 %). Měřítkem pro tato dvě kritéria jsou tzv. podpora a spolehlivost. V případě pravidel typu $A \Longrightarrow B$, kde A i B představují současný výskyt (konjunkci) příslušných položek, lze výše uvedená kritéria definovat následujícím způsobem:

$$\text{podpora}(A \Longrightarrow B) = \frac{n(A \wedge B)}{n} = P(A \wedge B)$$

$$\text{spolehlivost}(A \Longrightarrow B) = \frac{n(A \wedge B)}{n(A)} = \mathrm{P}(A|B),$$

kde n je počet všech trénovacích vzorů (transakcí) a $n(X)$ je počet takových vzorů (transakcí), které obsahují všechny položky zahrnuté v X. Podpora tedy odpovídá pravděpodobnosti, že náhodně zvolená transakce bude obsahovat všechny položky uvedené jak na levé straně příslušného pravidla, tak také položky uvedené na jeho pravé straně. Spolehlivost odpovídá skutečné přesnosti pravidla v případě, že daná transakce obsahuje všechny položky uvedené na levé straně tohoto pravidla. Dalším měřítkem kvality generovaných pravidel je tzv. zlepšení, jehož hodnota vyjadřuje, oč lepší je použít dané pravidlo než jeho závěr prostě předpokládat.

$$\text{zlepseni}(A \Longrightarrow B) = \frac{\mathrm{P}(A \wedge B)}{\mathrm{P}(A)\mathrm{P}(B)},$$

Pro $\text{zlepseni}(A \Longrightarrow B) \leq 1$, bude vést pravidlo k horším výsledkům než náhodná volba. Lepší výsledky mohou v takovém případě dávat pravidla typu $A \Longrightarrow \neg B$ s negovaným závěrem.

Jednou z nejpoužívanějších metod pro generování vhodných asociačních pravidel je algoritmus Apriori navržený R. Agrawalem [1]. Algoritmus Apriori generuje asociační pravidla ve dvou fázích. V první fázi algoritmus generuje všechny i-tice položek (atributů), které dosahují dostatečně vysoké podpory [2]. Tyto i-tice (někdy označované také jako časté kombinace položek) následně slouží pro generování pravidel, která dosahují alespoň předem stanovené meze spolehlivosti. Možné kombinace položek se generují prohledáváním možných kombinací do šířky. Při hledání kombinací délky i se přitom využívá znalosti kombinací délky $i-1$ a pro vytváření kombinace délky i budeme požadovat, aby všechny její podkombinace délky $i-1$ splňovaly předem stanovené požadavky na podporu, protože $\text{podpora}_{\text{nadkombinace}} \leq \text{podpora}_{\text{kombinace}}$.

Po nalezení kombinací, které vyhovují četností, je na jejich základě možné vytvářet asociační pravidla. Přitom se každá kombinace $Comb$ rozdělí na všechny možné dvojice podkombinací $Comb_A$ a $Comb_B$ takové, že

$$Comb_B = Comb - Comb_A$$

(za předpokladu, že $Comb_A \cap Comb_B = \emptyset$ a $Comb_A \wedge Comb_B = Comb$). Vytvářená pravidla typu

$$Comb_A \Longrightarrow Comb_B$$

budou mít podporu odpovídající četnosti kombinace $Comb$ a jejich spolehlivost bude odpovídat podílu četností kombinací $Comb$ a $Comb_A$.

Při generování častých kombinací položek (i-tic) je třeba projít celou databázi transakcí a vyhledat všechny položky s dostatečně vysokou podporou. Z nalezených $i-$tic položek se generují $(i+1)$-tice tak, aby všechny jejich podkombinace velikosti i měly dostatečně vysokou podporu. Přitom už ovšem není nutné procházet celou databázi, protože všechny i-tice s vysokou podporou už byly vybrány během předchozího průchodu. Dostatečná míra podpory pro vytvořené $(i+1)$-tice se ověří až následně během následujícího průchodu databází. V první fázi algoritmu Apriori je tedy pro každé i nutné jedenkrát projít databázi transakcí. Vzhledem k tomu, že se zvyšováním uvažovaného počtu položek i rychle klesá jejich podpora, bude množina vhodných $(i+1)$-tic brzy prázdná. V praxi bývá počet uvažovaných konjunkcí poměrně nízký – obvykle 2–3.

Algoritmus APRIORI

Krok 1: Do $Cast\_Kombinace_1$ přiřaď všechny položky, které dosahují alespoň požadované podpory.

Krok 2: Polož $k = 2$.

Krok 3: Dokud $Cast\_Kombinace_k \neq \emptyset$

Krok 3.1: Pomocí funkce APRIORI-GEN vygeneruj na základě $Cast\_Kombinace_{k-1}$ množinu kandidátů $Kand_k$.

Krok 3.2: Do $Cast\_Kombinace_k$ zařaď ty kombinace z $Kand_k$, které dosáhly alespoň požadované podpory.

Krok 3.3: Zvětši čítač k.

Funkce APRIORI-GEN($Cast\_Kombinace_{k-1}$)

Krok 1: Pro všechny dvojice kombinací $Comb_p$, $Comb_q$ z množiny $Cast\_Kombinace_{k-1}$

Krok 1.1: Pokud se $Comb_p$ a $Comb_q$ shodují v $k-2$ položkách, přidej $Comb_p \wedge Comb_q$ do $Kand_k$

Krok 2: Pro každou kombinaci $Comb$ z $Kand_k$

Krok 2.1: Pokud některá z jejich podkombinací délky $k-1$ není obsažena v $Cast\_Kombinace_{k-1}$, odstraň $Comb$ z $Kand_k$.

V druhé fázi jsou nalezené časté kombinace položek (i-tice) použité pro generování všech možných asociačních pravidel, u nichž je následně testována jejich spolehlivost. Začíná se se všemi pravidly, která mají na pravé straně jedinou položku a přitom mají dostatečně vysokou spolehlivost. Z těchto pravidel se pak vytvářejí pravidla, jejichž pravá strana obsahuje konjunkci dvou položek a rovněž zachovávají dostatečně vysokou spolehlivost. Celý proces inkrementálně pokračuje, dokud nejsou vyčerpána všechna možná pravidla pro rozšíření. Při generování pravidel ve druhé fázi algoritmu Apriori už není třeba přístup do databáze. Pro výpočet spolehlivosti je nutné znát jen podporu kombinací položek, které tvoří levou stranu pravidel. Jejich podpora už ale byla vypočtena a většinou i uložena např. v hashovací tabulce během prvního kroku algoritmu Apriori.

Protože jsou databáze, na jejichž základě se generují asociační pravidla, typicky veliké, snaží se některé z dalšách variant algoritmu Apriori urychlit proces učení omezením počtu prováděných průchodů databází redukcí počtu vytvářených pravidel, která je třeba testovat, a omezením doby potřebné pro výpočet podpory jednotlivých kombinací položek [6]. Počet průchodů databází lze redukovat kontrolou všech možných i-tic i $(i+1)$-tic během téhož průchodu databází. Možné $(i+1)$-tice se v takovém případě generují z (dosud) netestovaných i-tic, a proto bývá jejich počet mnohem větší, než je ve skutečnosti třeba. Z tohoto důvodu je daný přístup vhodný pouze pro úlohy s časově velmi náročnými průchody databází.

Další přístup, kterým lze dosáhnout podstatné redukce průchodů databází je vzorkování. Protože se asociační pravidla typicky extrahují z velkých databází, může vzorkování během jediného průchodu vytvořit reprezentativní sadu vzorů, kterou už je možné uchovávat v operační paměti. Tyto vzory se pak používají ke generování všech i-tic se zmírněnými požadavky na podporu. Jejich podpora je následně testována během dalšího průchodu databází a pouze ty i-tice, které mají dostatečně vysokou podporu budou zachovány. Tento přístup vyžaduje pouze dva průchody celou databází. Pouze velmi zřídka je potřebný ještě třetí průchod, při němž jsou odstraněny ještě takové i-tice s příliš nízkou podporou, jejichž všechny $(i-1)$-tice měly dostatečně vysokou podporu.

Při redukci doby potřebné pro výpočet podpory kombinací položek se využívá stromových struktur. Protože jsou atributy v množinách položek (i-ticích) uspořádané, lze generované množiny položek reprezentovat pomocí stromů, ve kterých má kořen anebo kterýkoliv vnitřní uzel tolik vycházejících hran, kolik existuje možných atributů v první části množiny položek. Každý podstrom se proto skládá ze všech množin položek, které sdílí ve své první části stejný atribut. Každá množina položek odpovídá listu stromu. Vždy když se načítá nový vzor z databáze, algoritmus prochází strom od kořene směrem k listům podle atributů vzoru. Jakmile je dosažen list, zvýší se jeho četnost o jedničku.

V praxi závisí výpočetní náročnost procesu extrakce asociačních pravidel také na prahových hodnotách stanovených pro podporu a spolehlivost. Obvykle

totiž vyhovuje znalost poměrně omezeného počtu nejlepších asociačních pravidel. V případě nové databáze však může být obtížné zvolit vhodné prahové hodnoty předem. Nevhodné nastavení prahů pak vede buď k prázdné množině asociačních pravidel, anebo naopak k příliš velkým množinám pravidel. Druhý případ lze obvykle poznat podle extrémně pomalých výpočtů a příliš dlouhých a nesrozumitelných pravidel. Tomu lze zabránit počátečním nastavením poměrně vysokých prahových hodnot pro podporu a spolehlivost. Proces učení se následně několikrát opakuje s pomalu klesajícími prahovými hodnotami pro podporu a spolehlivost, dokud se nevygeneruje dostatečně velká množina pravidel.

6 Závěr

Techniky určené pro automatické zpracování dat z otevřených zdrojů by měly být ze zřejmých důvodů co možná nejméně vázané na konkrétní typ dat (text, obrazová data, aj) a neměly by klást velké nároky na předzpracování. Zároveň by však měly podporovat rychlé zpracování typicky velkého množství dat. Měly by poskytovat spolehlivé výsledky, které jsou robustní vzhledem k možným nepřesnostem a chybám zaneseným v datech. Jako nejvhodnější se proto jeví metody založené na principu strojového učení.

Výhodou Bayesovských metod je intuitivní pravděpodobnostní interpretace řešených úloh a jejich výsledků. Nevýhodou obecných Bayesovských klasifikátorů je nutnost udržovat tabulky pravděpodobností pro všechny možné kombinace atributů. Jejich velikost přitom v obecném případě roste exponenciálně s počtem atributů. Tuto nevýhodu odstraňuje tzv. naivní Bayesovský klasifikátor, který navíc nevyžaduje ani příliš velkou trénovací množinu dat.

Umělé neuronové sítě představují matematické modely inspirované biologickou nervovou soustavou. Pro učení s učitelem se používají tzv. vrstevnaté neuronové sítě typu zpětného šíření, které se rychle učí a dobře zobecňují extrahované znalosti. Na rozdíl od Bayesovských modelů však nepodporují snadnou interpretaci získaných znalostí uložených typicky ve vahách sítě.

Na principu samoorganizace jsou založeny Kohonenovy mapy, které se využívají při analýze dat ve vstupním prostoru. Nevýhodou tohoto modelu je neefektivní proces rozpoznávání, který vyžaduje výpočet aktivace všech neuronů v dané síti. Problematické je rovněž použití této metody pro atributy s kategoriálními hodnotami, pro které není dobře definovaný koncept vzdálenosti.

Výhodou asociačních pravidel jsou jasné a srozumitelné výsledky prováděné analýzy. Metoda umožňuje zpracovávat data s variabilní délkou a bez znalosti požadovaných výstupů. Výpočetní nároky ovšem rostou exponenciálně s počtem uvažovaných položek (atributů v databázi). Je velmi obtížné stanovit adekvátní počet položek, některé významné avšak relativně řídké položky tak mohou mít omezenou podporu.

Literatura

[1] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A. I. Fast discovery of association rules, in *Fayyad et al. (eds): Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 1996.

[2] Berka, P. *Dobývání znalostí z databází.* Praha : Academia, 2003.

[3] Berry, M. J. A., Linoff, G. *Data Mining Techniques For Marketing, Sales, and Customer Support.* New York : John Wiley & Sons, 1997.

[4] Chakrabarti, S. *Mining the Web: Discovering Knowledge from Hypertext Data.* San Francisco, CA : Morgan Kaufmann, 2003.

[5] Kohonen, T. *Self-Organizing Maps.* Berlin : Springer-Verlag, 2001.

[6] Kononenko, I., Kukar, M. *Machine Learning and Data Mining: Introduction to Principles and Algorithms.* Chichester, UK : Horwood Publishing, 2007.

[7] Mena, J. *Investigative Data Mining for Security and Criminal Detection.* Butterworth Heinemann, 2003.

[8] Mitchell, T. M. *Machine Learning.* McGraw-Hill, 1997.

[9] Mrázová, I., Dagli, C. H. Semantic clustering of the World Bank data, *International Journal of General Systems*, DOI: 10.1080/03081070701210345, Taylor & Francis, pp. 1–23, 2007.

[10] Mrázová, I., Wang, D. Improved generalization of neural classifiers with enforced internal representation, *Neurocomputing*, vol. 70, No. 16–18, pp. 2 940–2 952, 2007.

[11] Rumelhart, D. E., Hinton, G. E., Williams, R. J. Learning representations by back-propagating errors, *Nature*, vol. 323, pp. 533–536, 1986.

[12] Samarasinghe, S. *Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition.* New York : Auerbach Publicationss, 2007.

Některé méně tradiční metody detekce škodlivých aktivit v IP sítích

Petr Břehovský

E-mail: breh@breh.cz

Abstrakt

Příspěvek je věnován popisu levných a snadno dostupných technologií detekce útoků a škodlivého kódu, založených na vizuálních metodách analýzy paketů a dedikovaných detekčních sítích.

Časy, kdy bylo možné ponechat v Internetu „nezáplatovaný“ počítač bez dozoru i po několik měsíců aniž by doznal závažnější újmy jsou nenávratně pryč. S masivním rozšířením automaticky se propagujících červů je současná životnost implicitních instalací běžných operačních systémů velmi nízká. U některých operačních systémů se pohybuje pouze v desítkách minut (http://isc.sans.org/survivaltime.html).

Pokud se nacházíme v prostředí Internetu, nezbývá nám než sledovat trendy, exponovaná zařízení pečlivě konfigurovat (zabezpečovat) a záplatovat a záplatovat. Zabývat se odchytem a studiem červů a dalších škodlivých kódů má snad pro nás pouze informační hodnotu (pokud nejsme výrobce antivirového software, IDS apod.).

Úplně jiná situace však nastává v prostředí privátních sítí (Intranetů). Červ v takovéto síti způsobí pravděpodobně hodně rozruchu a rozhodně nebude stačit, když o něm budeme pouze vědět. Jistě bude nutné identifikovat zdroje jeho šíření a následně ho odstranit. Obdobná situace nastává i v případě útoků realizovaných člověkem. Rozdíl je pouze v tom, že útok realizovaný lidským útočníkem bývá mnohem méně nápadný, než útok realizovaný robotem. V obou případech je však nutné útočníka detekovat, vystopovat a odstranit.

Tradiční metodou detekce útoků je použití IDS. Ten kdo IDS provozuje však ví, že tato zařízení jsou poměrně náročná na správu, produkují množství falešných poplachů, šikovného útočníka nemusí vůbec detekovat a jejich implementace a provoz může být velmi nákladná. Prvotní nadšení z nového IDS brzy pomine a velká očekávání do něj vkládaná mnohdy nejsou uspokojena. Implementace IDS totiž nespočívá pouze v jeho koupi a konfiguraci. Největší práce

začíná až při ladění pravidel, které mají případné útoky detekovat. Množství falešných poplachů je v počátečních fázích nasazení tak velké, že hrozí nebezpečí frustrace a rezignace jak implementátorů, tak týmu, který má nahlášené „incidenty“ řešit.

Napomoci v řešení této situace mohou dva základní principy:

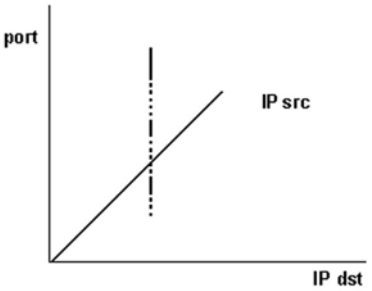
- rychlá orientace v datech přenášených sítí
- oddělení běžného provozu od škodlivých aktivit

Rychlá orientace v datech přenášených sítí

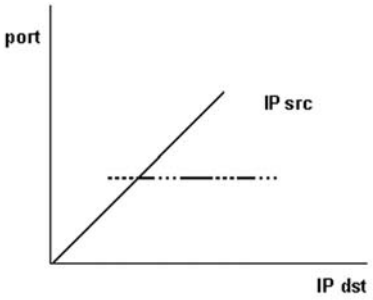
Množství a variabilita dat putujících po dnešních sítích je enormní. Pokud je chceme analyzovat za účelem hledání podezřelých aktivit, můžeme to provádět automatizovaně (IDS), nebo ručně. Oba přístupy mají své výhody i nedostatky. Automatizované (strojové) metody umožňují díky výpočetnímu výkonu analyzovat téměř každý přenesený bajt, ale chybí jim „nadhled“ a schopnost abstrakce. Člověk naproti tomu není v žádném případě schopen rychle a dopodrobna analyzovat větší množství dat, zato umí jediným pohledem dospět k pocitu, že „Tady safra něco nehraje!“. Metodou, jak mu umožnit takový pohled na data je jejich VIZUALIZACE.

Grafická reprezentace síťových dat

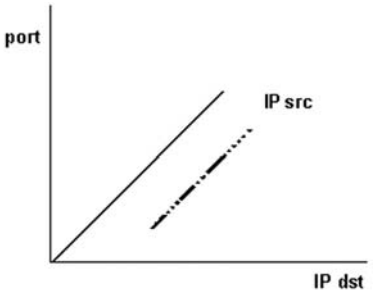
Zajímavou a jednoduchou myšlenkou je zobrazit přenášené pakety ve třech rozměrech tak, že například ose X budou odpovídat IP adresy monitorované sítě (destination), ose Z IP adresy externích sítí (source) a ose Y cílové porty. Jediným pohledem na takovou reprezentaci dat lze odhalit skenování portů:



útok červa:



i počítač a port, o který je „všeobecný zájem“ (DDOS útok, nebo vytížená služba):



Experimentálně je tato myšlenka implementována nástrojem Spinning Cube of Potential Doom (http://www.nersc.gov/nusers/security/TheSpinningCube.php).

Kód GPL implementace pro Linux lze najít na http://www.kismetwireless.net/doomcube/.

Další inspiraci v této oblasti může poskytnout RUMINT (http://www.rumint.org), nebo poněkud méně podporovaný NAV (https://www.cs.ubc.ca/~tmm/courses/cpsc533c-04-fall/slides/meghana.final.ppt).

„Srozumitelná“ reprezentace síťových dat

K transformaci dat vyskytujících se v síti do textové formy přijatelné pro člověka, zle použít například TCPFLOW (http://www.circlemud.org/~jelson/software/tcpflow/), který umí vyextrahovat vybraná TCP spojení do souborů,

```
-rw-r–r– 1 root root   218 Nov 20 15:36 010.000.000.139.33806-193.166.003.002.00021
-rw-r–r– 1 root root  3502 Nov 20 15:36 010.000.000.139.33922-212.071.133.148.00443
-rw-r–r– 1 root root  3684 Nov 20 15:36 193.166.003.002.14110-010.000.000.139.33818
```

```
-rw-r–r– 1 root root    957 Nov 20 15:36 193.166.003.002.18282-010.000.000.139.33849
-rw-r–r– 1 root root   1285 Nov 20 15:36 193.166.003.002.41919-010.000.000.139.33843
-rw-r–r– 1 root root 104328 Nov 20 15:36 193.166.003.002.43923-010.000.000.139.33856
-rw-r–r– 1 root root   4329 Nov 20 15:36 193.166.003.002.61648-010.000.000.139.33825
-rw-r–r– 1 root root   3684 Nov 20 15:36 193.166.003.002.62531-010.000.000.139.33815
-rw-r–r– 1 root root  29630 Nov 20 15:36 212.071.133.148.00443-010.000.000.139.33922
```

jejichž jména reprezentují IP adresy a porty spojení.

Samotné soubory pak obsahují informace o komunikaci aplikačním protokolem:

```
010.000.000.139.33806-193.166.003.002.00021:
USER anonymous
PASS breh
SYST
PASV
LIST
CWD pub
PASV
LIST
PASV
LIST
CWD doc
PASV
LIST
CWD martial-arts
PASV
LIST
CWD ../security
PASV
LIST
CWD worm
PASV
LIST
TYPE I
PASV
RETR gao_rpt.txt
QUIT
```

včetně přenášeného souboru:

```
193.166.003.002.43923-010.000.000.139.3385:
United States General Accounting Office
GAO                          Report to the Chairman, Subcommittee on
                             Telecommunications and Finance,
                             Committee on Energy and Commerce
                             House of Representatives
```

```
June 1989                       COMPUTER SECURITY

                                Virus Highlights Need
                                for improved Internet
                                Management

GAO/IMTEC-89-57
... kráceno ...
```

Podrobné technické informace o spojení nám poskytne TCPTRACE (http://jarok.cs.ohiou.edu/software/tcptrace/):

```
TCP connection 1:
        host a:        10.0.0.139:33806
        host b:        193.166.3.2:21
        complete conn: yes
        first packet:  Tue Nov 20 15:22:24.606891 2007
        last packet:   Tue Nov 20 15:26:00.037815 2007
        elapsed time:  0:03:35.430924
        total packets: 103
        filename: pokus

    a->b:                              b->a:
      total packets:          63         total packets:          40
      ack pkts sent:          62         ack pkts sent:          40
      pure acks sent:         35         pure acks sent:          3
      sack pkts sent:          0         sack pkts sent:          0
      dsack pkts sent:         0         dsack pkts sent:         0
      max sack blks/ack:       0         max sack blks/ack:       0
      unique bytes sent:     218         unique bytes sent:    3000
      actual data pkts:       26         actual data pkts:       35
      actual data bytes:     218         actual data bytes:    3000
      rexmt data pkts:         0         rexmt data pkts:         0
      rexmt data bytes:        0         rexmt data bytes:        0
      zwnd probe pkts:         0         zwnd probe pkts:         0
      zwnd probe bytes:        0         zwnd probe bytes:        0
      outoforder pkts:         0         outoforder pkts:         0
      pushed data pkts:       26         pushed data pkts:       35
      SYN/FIN pkts sent:     1/1         SYN/FIN pkts sent:     1/1
      req 1323 ws/ts:        Y/Y         req 1323 ws/ts:        Y/Y
      adv wind scale:          0         adv wind scale:          0
      req sack:                Y         req sack:                Y
      sacks sent:              0         sacks sent:              0
      urgent data pkts:        0 pkts    urgent data pkts:        0 pkts
      urgent data bytes:       0 bytes   urgent data bytes:       0 bytes
      mss requested:        1460 bytes   mss requested:        1460 bytes
```

```
max segm size:          18 bytes  max segm size:        1292 bytes
min segm size:           6 bytes  min segm size:          19 bytes
avg segm size:           8 bytes  avg segm size:          85 bytes
max win adv:          9044 bytes  max win adv:         50400 bytes
min win adv:          5840 bytes  min win adv:         50400 bytes
zero win adv:            0 times  zero win adv:            0 times
avg win adv:          8908 bytes  avg win adv:         50400 bytes
initial window:         16 bytes  initial window:        307 bytes
initial window:          1 pkts   initial window:          1 pkts
ttl stream length:     218 bytes  ttl stream length:    3000 bytes
missed data:             0 bytes  missed data:             0 bytes
truncated data:          0 bytes  truncated data:          0 bytes
truncated packets:       0 pkts   truncated packets:       0 pkts
data xmit time:    203.150 secs   data xmit time:    215.152 secs
idletime max:      55343.3 ms     idletime max:      55412.1 ms
throughput:              1 Bps    throughput:             14 Bps
```

Výše uvedené nástroje by měly sloužit jako inspirace k dalšímu samostatnému bádání...

Oddělení běžného provozu od škodlivých aktivit

Tento princip můžeme realizovat instalací samostatného počítače, který bude sloužit jenom k detekci útoků, nebo dokonce vytvořením celé sítě, která nebude obsahovat žádné provozní ani uživatelské zařízení a veškerý provoz, který se v ní vyskytne tak bude možné považovat za škodlivou aktivitu. Obecně jsou taková zařízení označována jako HONEYPOTY, resp. HONEYNETY

Následuje stručný přehled typů těchto systémů:

Honeypot

Je zařízení, které emuluje uzel sítě i s jeho službami, tj. tváří se jako běžný server, směrovač, pracovní stanice atd. a zároveň umožňuje studovat aktivity útočníka. V rámci provozní infrastruktury však žádné uživatelské služby neposkytuje.

Honeypot s nízkou interaktivitou

Je honeypot, který útočníkovi poskytuje omezené prostředí pro interakci. Emuluje pouze vnější pohled na systém (typ, verzi operačního systému a provozované služby). Zpravidla neumožňuje simulaci úplného ovládnutí služby a operačního systému, ale účinně detekuje pokusy o útoky na síťové služby. Implementace je relativně jednoduchá a levná. Honeypot s nízkou interaktivitou také představuje poměrně malé bezpečnostní riziko pro síť ve které je provozován.

Honeypot s vysokou interaktivitou

Emuluje služby a operační systém s vysokou věrností a umožňuje tak útočníkovi kompletní průnik až na úroveň operačního systému. Implementace a správa takového honeypotu je složitá, nákladná a představuje vyšší riziko zneužití útočníkem.

Darknet

Je část alokovaného a běžně směrovaného IP prostoru, ve kterém se však nenachází žádný aktivní server, nebo služba. Z hlediska uživatele je taková síť zcela prázdná. Ve skutečnosti Darknet obsahuje alespoň jeden server, který zachytává a analyzuje všechny pakety, které se v Darknetu mohou objevit. Protože se v Darknetu nemůže nikdy vyskytnout legitimní paket (neběží v něm žádná oficiální služba), jsou všechny pakety nacházející se v Darknetu výsledkem omylu, chybné konfigurace, nebo nekalé aktivity. Darknet tak výraznou měrou redukuje výskyt paketů způsobujících falešné poplachy. Darknet nejenom doplňuje IDS nacházející se v „živé“ části sítě (detekuje útoky a jejich zdroje), ale může podstatnou měrou přispět k jeho vyladění co se týče odlišování reálných útoků od falešných poplachů, obzvláště v případě skenů sítí a útoků malware.

Klientský honeypot

Umožňuje studovat WEB servery obsahující škodlivý kód, který je spuštěn aktivitou klienta (kliknutím na odkaz apod.). Tento kód nemůže být běžným honeypotem kvůli jeho pasivní podstatě detekován.

Shadow Honeypoty

Kombinují filtraci paketů, detekci anomálií a honeypoty ve snaze odstranit nedostatky těchto technologií projevující se pokud jsou implementovány samostatně. http://www.usenix.org/events/sec05/tech/full_papers/anagnostakis/anagnostakis_html/replay.html

Konkrétní implementace

Dnes již klasickým představitelem honeypotu s nízkou interaktivitou je HONEYD (http://www.honeyd.org/). Umožňuje implementovat na jediném počítači celé sítě virtuálních zařízení, která se tváří, že provozují rozličné operační systémy a služby. Honeyd umožňuje také simulovat síťové topologie s dedikovanými směrovači, latencí a simulovanými ztrátami paketů.

Konfigurace virtuálního serveru *Windows XP SP1 s WEB serverem a směrovače Cisco 1601R s IOSem 12.1* může vypadat například takto:

```
create windows
set windows personality "Microsoft Windows XP Professional SP1"
set windows uptime 1728650
set windows maxfds 3
set windows default tcp action reset
add windows tcp port 80 "sh /usr/share/honeyd/scripts/win32/web.sh"
add windows tcp port 22 "/usr/share/honeyd/scripts/ssh.sh"

create router
set router personality "Cisco 1601R router running IOS 12.1(5)"
set router default tcp action reset
add router tcp port 23 "/usr/share/honyed/scripts/router-telnet.pl"

bind 10.3.0.1 router
bind 10.3.1.1 router
bind 10.3.1.12 windows
```

K emulaci chování operačního systému je použit soubor otisků operačních systémů programu NMAP. Jednotlivé služby jsou definovány pomocí shell, nebo perl skriptů. Tyto skripty realizují interakci s útočníkem a logování všech akcí do definovaných souborů.

Běžné konfigurace honeyd se hodí k detekci pokusů o útoky. Těžko však už zachytí celý průběh útoku, nebo škodlivý kód červa. K detekci a hlavně zachycení celého kódu červa je mnohem lépe uzpůsoben NEPENTHES (http://nepenhes.mwcollect.org), který dokáže emulovat běžně známé bezpečnostní díry, díky kterým se červi šíří. Pomocí tohoto systému, který dokáže stejně jako honeyd realizovat více virtuálních počítačů na jednom fyzickém zařízení tak můžeme nejenom získat kód červa, ale také identifikovat zdroj útoku. Statistiky a informace získané z provozu Nepenthes v Internetu najdeme na adrese http://www.mwcollect.org.

Trochu jiný přístup k detekci a analýze útoků a škodlivého kódu představuje projekt DARKNET (http://www.team-cymru.org). Darknet je část alokovaného a běžně směrovaného IP prostoru, ve kterém se však nenachází žádný aktivní server, nebo služba. Z hlediska uživatele je taková síť zcela prázdná.Ve skutečnosti Darknet obsahuje alespoň jeden server, který zachytává a analyzuje všechny pakety, které se v Darknetu mohou objevit. Protože se v Darknetu nemůže nikdy vyskytnout legitimní paket (neběží v něm žádná oficiální služba), jsou všechny pakety nacházející se v Darknetu výsledkem omylu, chybné konfigurace, nebo nekalé aktivity. Darknet tak výraznou měrou redukuje výskyt paketů způsobujících falešné poplachy.

Minimální konfigurace Darknetu obsahuje směrovač (nebo alespoň jeho jedno rozhraní) a server, na kterém budeme provádět analýzu paketů vyskytujících se v Darknetu. Samozřejmě sem můžeme umístnit i další prvky, jako například jednu IDS sondu, která bude sloužit pro porovnávání dat z IDS sond z provozních částí sítě.

Směrovač by měl být dobře zabezpečen a musí směrovat veškerá data adresovaná do Darknetu na „naslouchací“ (viz dále) rozhraní serveru. Data odcházející z Darknetu mohou být blokována. Server má dvě síťová rozhraní. Jedno slouží k jeho správě (ssh, správa analyzátoru a SNMP) a na druhém je provozován síťový analyzátor. Vhodná je například kombinace TCPDUMP (http://www.tcpdump.org/) a ARGUS (http://www.qosient.com/argus/flow.htm).

Podrobně je konfigurace Darknetu popsána na výše zmíněném http://www.team-cymru.org.

Limity

Uvedené technologie mají samozřejmě i svá omezení. Jsou svou podstatou pasivní, takže nemohou analyzovat komunikaci, která probíhá mezi jinými uzly sítě.

Honeypot, nebo Darknet nemůže odhalit přesně cílený útok na konkrétní server, ani červa, který se propaguje pomocí tzv. hitlistu (předem známého seznamu zařízení obsahujících chybu, kterou červ zneužívá).

Samotné honeypoty může zkušený útočník (a možná o sofistikovaný červ) identifikovat a vyhnout se jim.

Závěr

Stávající příspěvek je zaměřen spíše na úhel pohledu správce interní sítě, který používá zmíněné metody jako doplňkové k IDS. Všechny výše uvedené nástroje lze ale s úspěchem použít (a pro některé uživatele je to i jejich hlavní využití) k detekci a analýze útoků zcela nezávisle na IDS. Jedná se hlavně o zkoumání škodlivého kódu a útočníků ve veřejných sítích (Internetu). Je možné nasadit je i zcela samostatně, ovšem s vědomím omezení, které mají.

Literatura

[1] *Spinning Cube of Potential Doom.*
Internet: http://www.nersc.gov/nusers/security/TheSpinningCube.php

[2] Provos, N. *A Virtual Honeypot Framework.*

[3] Baecher, P., Koetter, M., Holz, T., Dornseif, M., Freiling, F. *The Nepenthes Platform: An Eficient Approach to Collect Malware.*

[4] Team Cymru. *The Darknet Project.*
Internet: http://www.cymru.com/Darknet/

Praktické zkušenosti s implementací VoIP v síti O2

Petr Poupě

E-mail: petr.poupe@o2.com

Abstrakt

Příspěvek se bude zabývat nasazováním VoIP/IMS (IP Multimédia Subsystém) řešení v síti Telefónica O2 Czech Republic, a. s., se zaměřením na nesnáze a problémy s tím související.

Prezentaci v PowerPointu naleznete na adrese
http://www.europen.cz/Proceedings/32/

Implementace IP telefonie na Západočeské univerzitě v Plzni

Michal Petrovič

E-mail: petrovic@civ.zcu.cz

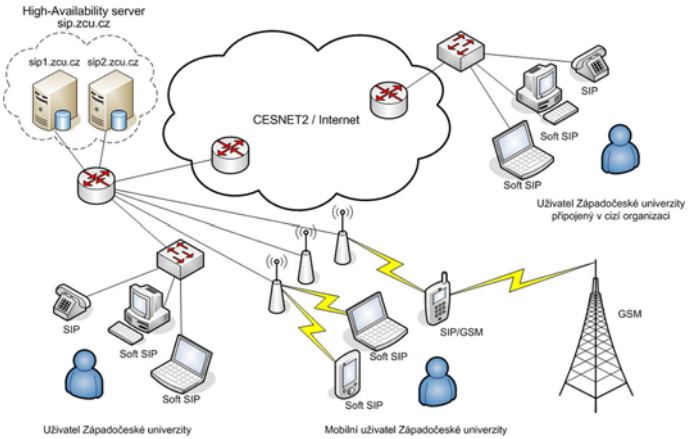
Cílem tohoto projektu bylo implementovat SIP IP telefonii na Západočeské univerzitě v Plzni a vyzkoušet její konvergenci do bezdrátového připojení. Služba SIP IP telefonie je určena nejen pro uživatele Západočeské univerzity, ale i pro cizí uživatele volající směrem do univerzity. Velikou výhodou této služby je její cena za volání, je totiž ve většině případů zadarmo.

Další obrovskou výhodou této služby je možnost volání odkudkoliv ze světa nebo i ze svého domova do univerzity nebo do spřátelené organizace za minimální poplatek případně rovnou opět zadarmo. Jelikož část našich uživatelů tvoří lidé cestující po celém světě na různé konference nebo jiné zahraniční stáže, šetří tato služba nemalé finanční prostředky vynaložené na komunikaci z libovolného místa (s IP konektivitou) v zahraničí směrem do naší organizace.

Základy této implementace vznikly již v říjnu 2006 zakoupením prvního serveru sloužící jako IP ústředna a jeho napojením na stávající telefonní síť Západočeské univerzity. V průběhu řešení toho projektu došlo k rozšíření počtu serverů na dva a tím i zvýšení dostupnosti poskytované služby IP telefonie při případném výpadku jednoho z nich.

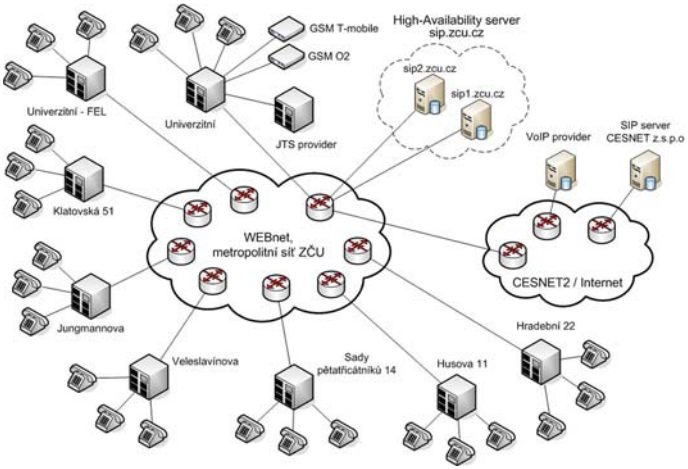
Služba SIP IP telefonie

Na obrázku 1 je znázorněná implementace služby SIP IP telefonie z pohledu uživatele. IP ústředna na ZČU je přímo propojená s IP ústřednou CESNET protokolem SIP a další napojení do této sítě je provedeno pomocí původní linky H323 přes gateway Cisco2651XM. Další možností komunikace z univerzitní IP telefonní sítě je pomocí TElephone NUmber Mapping (ENUM). Všechny hovory směřující ven z IP telefonní sítě ZČU jsou převedeny do mezinárodního formátu např. 420 377 638 888. Dále jsou tyto hovory dotazovány do lokálního privátního ENUM stromu ZČU, ENUM stromu nrenum.net – National Research and Education Networks (NREN), po té do stromu e164.arpa a následně směrovány na cíl dle výsledků těchto dotazů.



Obr. 1: Implementace SIP IP telefonie na Západočeské univerzitě – pohled uživatele

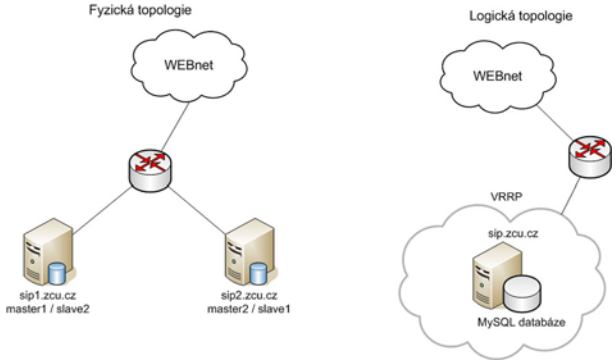
Napojení na stávající telefonní síť ZČU (ústředny Siemens) je pomocí již zmiňované gateway Cisco2651XM. Tato gateway je jediná možná cesta propojení pomocí protokolu IP. Další možností propojení IP telefonní sítě s původní telefonní sítí by mohlo být přímou ISDN linkou. Toto řešení však nebylo zvoleno, jelikož by obnášelo nákladné pořízení ISDN karty do ústředny Siemens. Celkový pohled na IP telefonní a telefonní síť je na obrázku 2.



Obr. 2: Celkový pohled na IP telefonní a telefonní síť ZČU

Redundantní řešení IP ústředny

Pro zvýšení dostupnosti a spolehlivosti celého VoIP systému je IP ústředna tvořená fyzicky dvěma servery (sip1.zcu.cz a sip2.zcu.cz), který tvoří dostatečně redundantní celek (sip.zcu.cz). Topologie tohoto řešení je vidět na obrázku 3. Celého řešení se skládá ze tří částí.



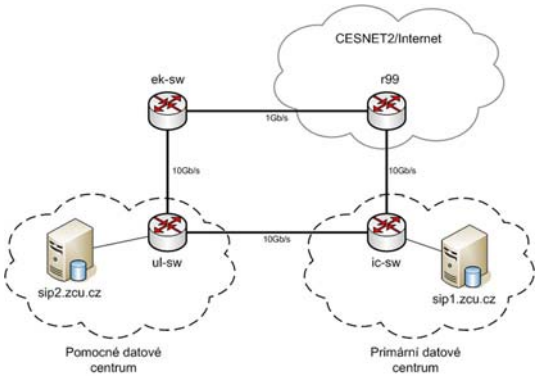
Obr. 3: Redundantní řešení IP ústředny

První částí je redundance IP připojení/provozu pomocí protokolu VRRP – Virtual Router Redundancy Protocol. Tento protokol pracuje na síťové vrstvě (IP adresy) a vytváří skupinu dvou nebo více IP adres, které po dohodě mezi sebou vystupují jako jedna virtuální adresa. Tuto virtuální adresu má přiřazenu vždy jeden stroj (master) a v případě jeho výpadku převezme virtuální adresu další funkční stroj s druhou nejvyšší prioritou (slave). Tímto je zajištěná redundantnost na úrovni IP vrstvy.

Druhou částí je replikace databáze MySQL typu MASTER-MASTER, ve které má IP ústředna uložena všechna potřebná data pro provoz. Při použití této replikace dochází k synchronizaci databáze vždy, když se na jednom ze serverů v databázi cokoliv změní.

Celé řešení tedy funguje tak, že na obou serverech běží aplikace OpenSER a její data jsou lokálně ukládána do databáze MySQL. Jeden server má vyšší prioritu a je tedy tzv. master, což znamená, že všechny pakety směrované na virtuální adresu sip.zcu.cz dojdou na něj a aplikace OpenSER je zpracuje. Jelikož MySQL databáze je replikovaná na druhý server, je zajištěn stejný obsah databáze na obou serverech. Při výpadku master serveru přebírá virtuální adresu slave server a všechny pakety směrované na virtuální adresu dojdou na něj. Opět zde běží aplikace OpenSER, která má stejný obsah databáze jako měl master server před výpadkem a pakety zpracuje. Po obnovení funkčnosti master serveru dojde k synchronizaci dat v databázi a opětovné převzetí virtuální adresy.

Poslední třetí část redundance je vhodné umístění serverů a zajištění jejich konektivity. První server (master) je umístěn v primárním datovém centru UI-420 budovy CIV. Druhý server (slave) je umístěn v pomocném datovém centru UL-008 laboratorního objektu. Obě tato datová centra mají na sobě nezávislý napájecí okruh, záložní zdroje napájení a konektivitu do zbytku počítačové sítě WEBnet. Jelikož jsou oba servery na stejné podsíti (požadavek protokolu VRRP), je dále zajištěna mezi těmito datovými centry redundantnost výchozí brány této podsítě pomocí protokolu HSRP – Hot Standby Router Protocol. Celkový pohled na redundantní řešení IP ústředny je na obrázku 4.



Obr. 4: Celkový pohled na redundantní řešení IP ústředny

V případě výpadku elektrické energie nebo poruše páteřního směrovače na jednom z datových center službu SIP IP telefonie přebere druhý server díky protokolu VRRP. Protokol HSRP zajistí stálý přístup do počítačové sítě WEBnet i v případě nedostupnosti druhého datového centra.

Hybridní IP telefonie

Jednou z nejzajímavějších částí celého VoIP systému ZČU je hybridní IP telefonie. Jedná se o IP telefonování pomocí tzv. hybridního IP telefonu v budovách Západočeské univerzity a jejich blízkého okolí, kde máme vybudovanou rozsáhlou bezdrátovou síť čítající přes 90 přístupových bodů.

Při implementaci hybridní IP telefonie byl kladen velký důraz na pohodlí uživatelů a technické parametry bezdrátové (Wi-Fi) části přístroje. Tímto je myšleno hlavně snadné zacházení s hybridním IP telefonem a transparentní chování telefonu při volání. Dále byly vyžadovány takové technické parametry bezdrátové části přístroje, které umožní jeho připojení do sítě eduroam stejně pojmenovaného mezinárodního projektu eduroam (http://eduroam.cz).

Zde se asi nejvíce osvědčily hybridní IP telefony řady E společnosti Nokia s operačním systémem Symbian. Tyto telefony umožňují zapnutí režimu „automaticky volat přes IP, pokud je dostupný Wi-Fi signál“ a zároveň umožňují, pokud si to uživatel přeje, jednoduchým způsobem bez jakékoliv změny nastavení volat pomocí sítě GSM (obrázek 5). Připojení do sítě eduroam u této řady přístrojů nebyl žádný problém a veškeré parametry připojení bylo možno přehledně nastavit.

Obr. 5: Ukázka volby možnosti volání na hybridním IP telefonu s OS Symbian

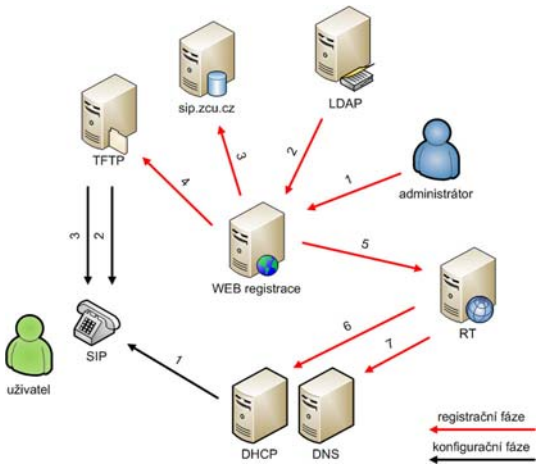
Obr. 6: Ukázka volání na hybridním IP telefonu s OS Windows Mobile 6

Dalším již však méně vhodným hybridním IP telefon byl S710 (Vox) firmy HTC s operačním systémem Windows Mobile 6 (WM6). Tento operační systém na rozdíl od svých předchůdců (Windows Mobile 5 a starší verze) má již integrovanou podporu IP telefonie. Nicméně způsob integrace a pohodlí (transparentnost) IP volání je zde o řád nižší než je tomu u přístrojů Nokia řady E. Operační systém WM6 umožňuje pouze nastavení „volat přes IP vždy“, „volat přes IP, když není dostupná síť GSM“ nebo „nikdy nevolat přes IP“. Dramaticky zde tedy chybí volba vynuceného volání přes GSM, pokud je přístroj nastaven v režimu „volat přes IP vždy“. Ukázka volání přes IP je na obrázku 6.

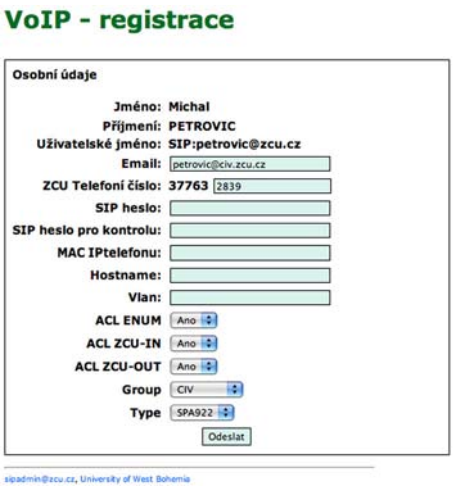
U operačních systémů jiných než WM6 a Symbian na telefonních přístrojích firmy Nokia řady E není žádná integrace IP telefonních hovorů. Je zde jediná možnost a to instalace dalšího programového vybavení v podobě různých komerčních i nekomerčních SIP klientů. Při použití těchto klientů je však velmi často špatně provázané IP volání s operačním systémem a tím i navýšená složitost obsluhovat tento telefon. Například pokud chce uživatel volat pomocí IP, musí si před tímto voláním spustit SIP klienta, zde provést volbu čísla a volat. Když IP hovor skončí a chtěl by volat pomocí sítě GSM musí tohoto klienta opět vypnout nebo přepnout na pozadí operačního systému.

Autokonfigurační systém

Autokonfigurační systém je určen k instalaci některých typů IP telefonů značky Linksys řady 9xx. Díky tomuto systému je možné zajistit instalaci i většího množství IP telefonů s minimálními náklady na čas administrátorů, kteří by museli dané telefony konfigurovat. Díky rozšíření funkčnosti IP telefonů od firmwaru 5.1.9 je možnost využít parametru v odpovědi od DHCP serveru a tím poskytnout IP telefonu informaci o TFTP serveru. Na tomto TFTP serveru může být umístěná výchozí konfigurace odkazující na specifickou konfiguraci pro daný IP telefon. Celý autokonfigurační systém vychází z administrátorského WEB rozhraní, kde se na základě vyplnění krátkého formuláře generuje konfigurační soubor na TFTP server a vytváří registrace daného uživatele. Závěrečnou fází je poslání žádosti o registraci doménového jména a IP adresy telefonu do systému Request Tracker (RT).



Obr. 7: Princip autokonfiguračního systému



Obr. 8: Administrátorské WEB rozhraní pro regisraci

Při vstupu na administrátorský WEB rozhraní (obrázek 8) je nutné nejprve zadat přihlašovací jméno uživatele (Orion login), kterého administrátor registruje do systému a chce pro jeho IP telefon poskytnout možnost autokonfigurace. Dále následuje krátký a jednoduchý formulář, kde je již většina dat před-vyplněná. Toto před-vyplnění je získáno na základě zadaného přihlašovacího jména uživatele a vyhledáním všech potřebných dat o tomto uživateli na serveru LDAP. Administrátor tedy vyplňuje pouze heslo pro SIP účet uživatele a fyzickou adresu (MAC) IP telefonu. Další položky jsou již volitelné a v návaznosti na ostatní systémy je vhodné je vyplnit. Jedná se o doménové jméno (hostname), číslo virtuální sítě (VLAN), možnosti cílového místa volání (ACL), skupiny uživatelů (Group) a typ IP telefonu (Type).

Základním stavebním kamenem celého autokonfiguračního systému je úprava konfigurace DHCP serveru. Zde je nutné do konfigurace přidat volitelnou položku pro odpověď tzv. DHCP Options. Tato rozšiřující položka má číslo 66 a obsahuje IP adresu TFTP serveru. Například v operačním systému Linux tato konfigurace může být umístěna v souboru dhcpd.conf a vypadat takto:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
option tftp-server-name "192.168.1.1";
...
}
```

TFTP server musí být nastaven, aby poslouchal a přijímal data na portu 69 protokolu UDP. V kořenovém adresáři TFTP serveru je nutné umístit výchozí

konfigurační soubor, kterým IP telefon nastavíme, aby si stáhnul svojí specifickou konfiguraci. Výchozí konfigurační soubor musí mít jméno ve tvaru *spa<typ>.cfg*, tedy například pro typ Linksys SPA922 by měl soubor jméno *spa922.cfg*. IP telefon si svojí specifickou konfiguraci vybere dle své MAC adresy. Formát zápisu MAC adresy je *$MA*. Například tedy pokud chceme načítat specifickou konfiguraci pro IP telefon s MAC 001122334455, pojmenujeme konfigurační soubor *spa001122334455.cfg* a ve výchozí konfiguraci uvedeme */spa$MA.cfg*. MAC adresu IP telefonu lze i zapsat v jiném formátu. Při uvedení ve výchozí konfiguraci */spa$MAC.cfg* bude IP telefon stahovat soubor *spa00:11:22:33:44:55.cfg*.

Obsah výchozího souboru:

```
<flat-profile>
<Profile_Rule ua="na">
   http://sip.zcu.cz/provisioning/config/spa$MA.cfg</Profile_Rule>
<Resync_Periodic ua="na">5</Resync_Periodic>
</flat-profile>
```

Celý autokonfigurační systém spolupracuje s IP ústřednou OpenSER verze 1.3. Tato ústředna, jak již bylo zmíněno, má svojí konfiguraci uloženou v MySQL databázi. Pokud tedy administrátor odešle vyplněný formulář, autokonfigurační systém vytvoří účet pro daného uživatele v MySQL databázi a vytvoří specifický konfigurační soubor. Takto lze autoregistrační systém napojit na jakoukoliv IP ústřednu, která má uživatelské účty uložené v MySQL databázi. Samotné vytváření specifického konfiguračního souboru pro daného uživatele je pomocí šablony, která obsahuje veškerou požadovanou konfiguraci IP telefonu. Při generování se do šablony vyplní uživatelská data a výsledek se uloží s požadovaným názvem souboru ve formátu XML.

Poslední součástí autokonfiguračního systému je vygenerování žádosti do systému Request Tracker. Administrátor DNS systému a DHCP serveru na základě této žádosti nakonfiguruje v DHCP serveru přidělování odpovídající IP adresy pro MAC adresu IP telefonu. Dále provede změny v systému DNS, aby správně fungoval překlad IP adresy IP telefonu na odpovídající doménové jméno. Tato žádost je vytvořena tak, aby systém Request Tracker označil za žadatele právě zaregistrovávaného uživatele. Po uzavření žádosti je zaslána informace žadateli o jeho registraci v systému VoIP, DNS a DHCP serveru.

Co umí souborové systémy

Jan Kasprzak

E-mail: kas@fi.muni.cz

Abstrakt

Souborový systém je jednou ze základních komponent UNIXového počítače. V této přednášce shrneme historii souborových systémů v UNIXu i Linuxu, představíme souborové systémy, které se reálně používají v současnosti v Linuxu (ext3, XFS, JFS, ReiserFS, JFFS2, OCFS2, GFS2), popíšeme pokročilé a nestandardní vlastnosti některých souborových systémů, a ukážeme, kam směřuje vývoj, včetně představení projektů, které se vize budoucnosti snaží realizovat: ext4, Reiser4, CRFS/BTRFS, POHMELFS, UBIFS a některé další.

Abstract

The File system is one of the principal components of the UNIX computer. In this paper, we will summarize a history of the file systems in both UNIX and Linux, we will introduce the file systems currently in use in Linux (such as ext3, XFS, JFS, ReiserFS, JFFS2, OCFS2, GFS2), we will describe advanced features of some file systems, and we will show possible directions of future development in this are: ext4, Reiser4, CRFS/BTRFS, POHMELFS, UBIFS, and some others.

Klíčová slova: Linux, storage, file system

1 Základní služby souborových systémů

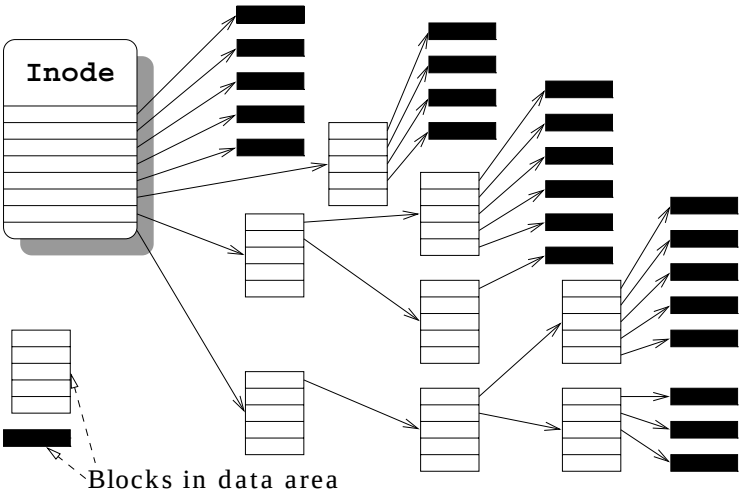
Operační systém UNIX, stejně tak jako jeho ideový následník Linux, používá k ukládání dat na diskové jednotky datovou strukturu, nazývanou *souborový systém* (file system, FS). Úlohou souborového systému je zavést na blokovém zařízení jako je pevný disk, disketa, či dnes stále více rozšířená flash paměť strukturu, která umožní využívat blokové zařízení jako úložiště stromu souborů a adresářů.

Takovéto struktuře říkáme také *metadata* (na rozdíl od *dat*, což už jsou uživatelem uložené informace uvnitř souborů). Jednotlivé souborové systémy se pak liší zejména tím, jak vypadají jimi používaná metadata, ale také tím, jakým způsobem s metadaty pracují.

Souborové systémy v UNIXu jsou ukládány na nezávislých blokových zařízeních (blokové zařízení je typicky oblast – *partition* – pevného disku) tak, že každé blokové zařízení se souborovým systémem na něm tvoří samostatný strom souborů a adresářů, tak zvaný *svazek* (volume). Jednotlivé svazky jsou pak spojeny do jednoho stromu souborů a adresářů, který je zpřístupněn procesům, běžícím na UNIXovém počítači.

1.1 I-uzly

Ve všech UNIXových souborových systémech se v různých podobách vyskytuje datová struktura nazývaná *i-uzel* (identifikační uzel, i-node). Tato struktura obsahuje metadata o jednom konkrétním souboru a je tedy jakousi „hlavičkou“ souboru. I-uzel je identifikován svým číslem, které je v rámci jednoho svazku jednoznačné. V i-uzlu jsou uložena přístupová práva souboru, vlastník a skupina souboru, typ souboru (běžný soubor, adresář, roura a další), časy (modifikace souboru, přístupu k souboru a modifikace i-uzlu), počet odkazů, délka souboru, a také odkaz na datové bloky souboru.



Obr. 1: I-uzel standardního UNIXového souborového systému

Datové bloky bývají z i-uzlu odkazovány různě. Tradiční UNIXové souborové systémy používají pseudologaritmickou strukturu naznačenou na obrázku 1:

přímo v i-uzlu je třináct ukazatelů na datový blok. Prvních deset ukazatelů ukazuje přímo na prvních deset bloků souboru. Jedenáctý ukazatel je tzv. první nepřímý odkaz – odkaz na blok v datové oblasti, který je celý rozdělen na ukazatele na skutečné datové bloky souboru. Dvanáctý ukazatel funguje podobně, jen bloky jsou odkazovány přes dvě úrovně. Třináctý ukazatel je pak třetí nepřímý odkaz.[1]

Tato struktura má několik zajímavých vlastností:

- Krátké soubory a začátky dlouhých souborů maji menší režii a přístup k nim je rychlejší.
- Na druhé straně adresa kteréhokoli bloku souboru je zjistitelná pomocí nejvýše tří přístupů na disk.
- K přístupu ke kterémukoli datovému bloku souboru není třeba nejprve načítat předchozí bloky souboru (lze tedy efektivně provádět i nesekvenční, náhodný přístup k souboru).
- Je-li v i-uzlu odkazovaný nějaký datový blok, neznamená to, že nutně musí být odkazovány všechny předchozí bloky. V tom případě mluvíme o *souboru s dírou* (sparse file). Takovéto soubory mohou za jistých okolností šetřit místo na disku. Díra se při čtení tváří jako bloky nulových bajtů, při zápisu se teprve alokuje místo na disku.[2]

1.2 Adresáře

Adresář (*directory*) je v UNIXu v podstatě běžný soubor, jen má u sebe příznak, že jde o adresář. Obsah tohoto souboru je interpretován jako seznam dvojic: pro každý soubor je zde jméno souboru a číslo i-uzlu, který je pod tímto jménem dostupný. Takto zejména může být jeden soubor (i-uzel) dostupný pod více jmény, případně z více adresářů na tomtéž svazku (tzv. *pevný odkaz*, hard link).

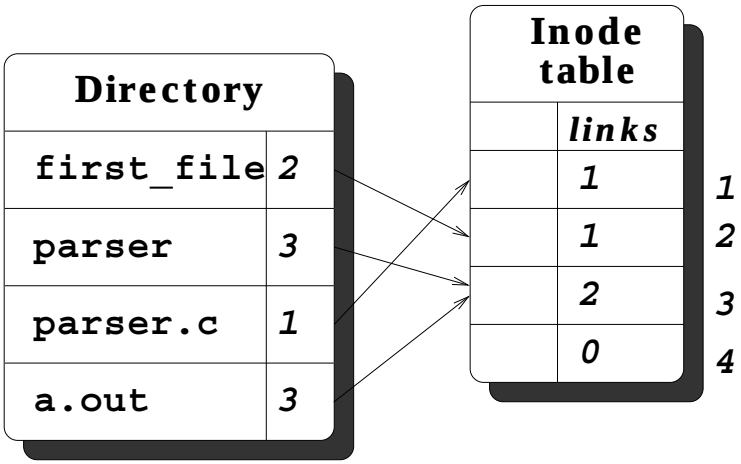
Každý adresář obsahuje položku „.“ (odkaz sám na sebe) a položku „..“ – odkaz na nadřazený adresář (v kořeni svazku odkaz sám na sebe).

Jméno souboru se v UNIXu interpretuje s rozlišováním velikosti písmen a může obsahovat libovolné znaky kromě lomítka (používá se pro oddělení komponent jména v rámci cesty) a nulového bajtu (používá se pro ukončení řetězců

[1] Máme-li například bloky velikosti 1 KB a 32-bitové (čtyřbajtové) ukazatele na blok, pak v i-uzlu je prvních 10 bloků (10 KB) odkazováno přímo, dalších 1 KB/4 B = 256 bloků odkazováno přes první nepřímý ukazatel, 256^2 bloků (tedy 64 MB) přes druhý nepřímý ukazatel a 256^3, čili 16 GB přes třetí nepřímý ukazatel. Maximální velikost souboru v takovémto souborovém systému je tedy o něco více než 16 GB.

[2] Příkladem souboru s dírou může být soubor `lastlog` (obvykle `/var/log/lastlog`). Vyzkoušejte na něm příkaz `ls -ls` – tento příkaz vypíše počet skutečně alokovaných bloků souboru i velikost souboru (offset posledního platného bajtu).

v jazyce C). Starší souborové systémy povolovaly jména souborů do délky 14 znaků, dnešní mají limit někde okolo 256 znaků. UNIX neeviduje kódování (znakovou sadu) jmen souborů, na dnešních systémech se obvykle používá UTF-8 pro svoji kompatibilitu s ASCII i univerzálnost.

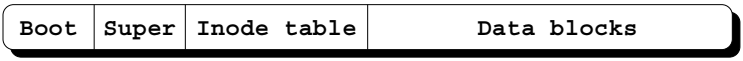


Obr. 2: Struktura adresáře

2 Historie

2.1 System V File System

Souborový systém z UNIXu System V (`s5fs`) může posloužit jako příklad toho, jaké minimální vlastnosti dnešní UNIXový souborový systém potřebuje.



Obr. 3: Struktura souborového systému `s5fs`

S5FS nechával na začátku volný blok (tzv. *boot block*) pro zavaděč systému. Dále následoval *superblok* se sumárními informacemi o tomto souborovém systému. Pak tabulka i-uzlů a nakonec datové bloky. Vadné bloky souboru byly znepřístupněny tak, že byly alokovány pro některý vyhrazený i-uzel, který nebyl odkazován z žádného adresáře. Volné datové bloky byly ukládány jako zřetězený seznam, což po delším používání souborového systému vedlo k fragmentaci, a tím ke sníženému výkonu. Typická velikost bloku S5FS je 512 bajtů nebo 1 KB.

2.2 FAT

Souborový systém FAT (*File Allocation Table*) vlastně nemá s UNIXem ani Linuxem nic společného, i když Linux i většina dalších systémů s ním umí pracovat. Uvádíme jej jako ilustraci jiného než UNIXového přístupu k souborovým systémům.

Obr. 4: Struktura souborového systému FAT

Souborový systém FAT začínal zaváděcím sektorem, dále byly dvě kopie alokační tabulky souborů a pak místo na datové bloky. V prvním datovém bloku začínal kořenový adresář. Alokační tabulka byla ve dvou kopiích z důvodu vyšší tolerance k výpadku. Alokační tabulka obsahovala jeden záznam (12-, 16- nebo 32-bitový) pro každý blok souborového systému (*cluster* v terminologii FAT). Tento záznam říkal, jestli je daný blok volný, vadný, nebo alokovaný v souboru nebo adresáři. V posledním jmenovaném případě pak záznam obsahoval číslo bloku, který po tomto bloku v daném souboru následuje (nebo informaci, že tento blok je posledním blokem daného souboru).

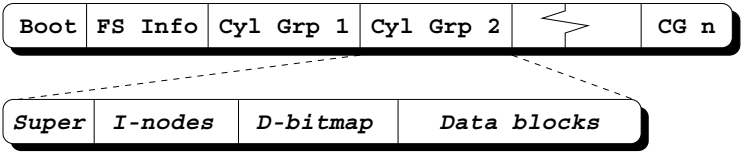
Adresář ve FAT obsahoval záznamy pevné délky (16 bajtů) se všemi potřebnými metadaty o souborech: 11 bajtů jména souboru interpretovaných jako osm znaků jména a tři znaky přípony bez ohledu na velikost písmen, čas modifikace souboru s přesností na dvě sekundy, odkaz na první datový blok souboru (další bloky se vyhledávaly pomocí alokační tabulky), atributy souboru (soubor nebo adresář, skrytý soubor, soubor jen pro čtení, atd.).

Pro kódování jmen souborů se v našich krajích používala kódová stránka IBM CP852 (i v systému Windows, který jinak data v souborech ukládal ve znakové sadě Windows-1250).

2.3 UFS

Souborový systém UFS (na některých systémech též FFS nebo EFS) pochází z BSD UNIXu. Autoři se snažili odstranit některé nedostatky S5FS. Celý souborový systém je rozdělen na několik úseků – *cylinder group*, CG (podle geometrie disku) – které obsahují kopii superbloku pro případ poškození primárního superbloku, část tabulky i-uzlů, nově bitmapu volných datových bloků a část datových bloků samotných.

Alokační strategií je zajištěno, že v případě volného místa nevzniká fragmentace ani v případě, že se paralelně vytváří více souborů (každý je pak vytvářen uvnitř jedné CG). Pro zvýšení výkonu se UFS typicky vytváří s většími bloky (4 nebo 8 KB). Aby se pak neplýtvalo místem na malých souborech, umožňuje



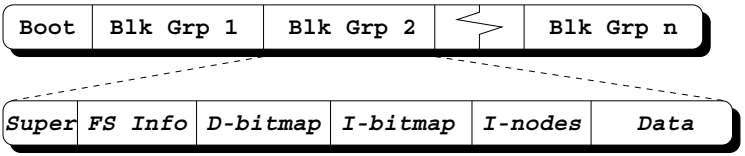
Obr. 5: Struktura souborového systému UFS

UFS uložit malý soubor do ne zcela využitého bloku na konci většího souboru (v terminologii UFS se tomuto říká *fragmenty*).

UFS původně používal synchronní zápis metadat, což v případě havárie počítače umožnilo programu `fsck` pro kontrolu disků hledat jen některé typy nekonzistencí a tím být rychlejší. Na druhé straně se v případě havárie v okamžiku zvětšování souboru mohlo stát, že informace o zvětšení je zanesena v metadatech, ale ještě ne v datech souboru, a že tedy soubor obsahuje nepřemazané bloky původně přidělené v jiných souborech (bezpečnostní problém). Novější implementace UFS umožňují asynchronní zápis metadat nebo asynchronní zápis metadat se zachováním pořadí operací (*soft updates*).

2.4 Ext2FS

Ext2FS (second extended filesystem) je souborový systém v Linuxu. Strukturou je podobný UFS – jen jednotlivé části se zde jmenují *block groups* a nejsou zarovnány podle fyzické geometrie disku jako u UFS (ostatně u dnešních disků se zónovým zápisem ani informaci o fyzické geometrii nemáme), ale tak, aby pokud možno jednotlivé datové struktury přesně zaplnily celočíselný počet diskových bloků. Oproti UFS je zde navíc bitmapa volných i-uzlů, která slouží k rychlejšímu nalezení volného i-uzlu při vytváření nového souboru.



Obr. 6: Struktura souborového systému Ext2FS

Ext2FS neimplementuje fragmenty a tento problém řeší opačným směrem: vytváří se s menší velikostí bloku (dříve 1 KB, dnes obvykle 4 KB) s tím, že při zápisu do souboru se bloky alokují po osmi (jsou-li k dispozici) a při uzavření souboru se nevyužité bloky uvolní. Tím se dosahuje podobné fragmentace jako u UFS s osminásobně většími bloky.

Mezi zajímavé vlastnosti Ext2FS patří tzv. rychlé symbolické linky – text symbolického linku, je-li kratší než 64 bajtů, je uložen přímo v i-uzlu na místě odkazů na datové bloky a dalších položek, které se u symbolických linků nevyužívají. Dále pak nové atributy souborů: nezměnitelný soubor, soubor pouze pro připojování dat na konec a bezpečné smazání souboru. Svazek může být připojen s asynchronním nebo synchronním zápisem metadat. Je možno zapnout nebo vypnout aktualizaci časů přístupu k souborům (časy modifikace se aktualizují i nadále).

3 Souborové systémy v Linuxu v současnosti

Většina současných souborových systémů přidává další vlastnosti: žurnálování (viz dále), access control lists (ACL), rozšířené atributy (například pro uložení bezpečnostního kontextu pro SELinux), často i on-line zvětšení svazku, například existuje-li nad LVM[3].

3.1 Tolerance k výpadku

Jedním z hlavních problémů starších souborových systému byla špatná tolerance k výpadku. I když program `fsck` je schopen nekonzistence opravit, jeho spuštění stojí čas a prodlužuje dobu výpadku. Souborové systémy se snaží podobným problémům předcházet podobným způsobem jako databáze – transakčním zpracováním. Uvažované změny jsou nejprve zapsány do logu (žurnálu) a teprve až je celá operace zapsána na disku v žurnálu, dochází ke změnám vlastních dat. V případě výpadku stačí jen přehrát ty transakce v žurnálu, které jsou zde zapsány jako kompletní.

Ani žurnálování ovšem nečiní program `fsck` zbytečným. V případech jako jsou hardwarové problémy a chyby operačního systému mohou i nadále vznikat nekonzistence souborového systému. Dostatečně robustní program pro kontrolu souborového systému je tedy i nadále podstatným kritériem výběru souborového systému.

Žurnálované souborové systémy obvykle transakčně zpracovávají jen operace nad metadaty. Důsledkem toho může být bezpečnostní problém podobného stylu, jako byl popsán u souborového systému UFS. Některé souborové systémy umožňují žurnálovat i operace nad daty, což ovšem s sebou nese větší režii. Jen velmi málo souborových systémů podporuje zlatou střední cestu – *uspořádané* (ordered) operace. V tomto režimu jsou zápisové operace nad daty vyřizovány asynchronně, ale teprve po jejich dokončení je do žurnálu ukončena transakce nad metadaty. Data samotná se tedy zapisují jen jednou (takže nedochází k vý-

[3] Logical Volume Manager

znamnému zpomalení proti pouhému žurnálování metadat), ale nemáme zmíněný bezpečnostní problém.

3.2 Ext3FS

Souborový systém Ext3FS je zpětně kompatibilním rozšířením Ext2FS. Nejvýznamnější změnou je žurnálování. Vývoj Ext3FS trval poměrně dlouho, nicméně jeho výsledkem je i obecná vrstva JBD (journalled block device), kterou používají i některé další souborové systémy a komponenty jádra. Žurnál může být umístěn i mimo vlastní blokové zařízení (například v baterií zálohované paměti RAM). Při korektním odpojení svazku (bez nedokončených transakcí v žurnálu) může být Ext3 svazek připojen jako Ext2.

Mezi další vlastnosti patří tzv. *sparse superblocks* – superblok není v každé block group, ale jen v exponenciálně narůstajícím intervalu BG, což sníží počet kopií superbloku na velkých svazcích a tím sníží dobu potřebnou pro připojení svazku i pro kontrolu svazku.

Adresáře v Ext3 mohou být uloženy jako lineární seznam nebo jako B-strom (což je efektivnější pro extrémně velké adresáře).

Ext3 implementuje všechny tři výše popsané režimy žurnálování s tím, že implicitní je *ordered* režim.

3.3 XFS

Do Linuxu se dostala implementace souborového systému XFS od SGI. Jedná se o plně 64-bitový žurnálovaný souborový systém. Svazek je rozdělen na oblasti velké obvykle 0,5 GB až 4 GB, nazývané *allocation groups* (AG). Informace v rámci AG jsou ukládány ve formě B+ stromu.

Při vytváření svazku lze uvést (případně `mkfs` zjišťuje sám), z kolika fyzických disků se skládá blokové zařízení a tuto informaci uloží do superbloku. Pomocí ní se pak v jádře počítá možná paralelizace diskových operací.

Asi nejzajímavější vlastností XFS je odložená alokace: většina souborových systémů používá odložený zápis dat (write-back). XFS toto posunuje ještě dále: u zapisovaných dat se i jejich *umístění* určuje až v okamžiku zápisu na disk. To umožní souborovému systému spojit více zápisových operací do jedné a dále tak snižovat fragmentaci. Na druhé straně toto komplikuje případnou implementaci *ordered* režimu pro žurnálování, takže XFS podporuje jen ostatní dva režimy. I zde je možno žurnálovat na externí zařízení.

XFS nepoužívá strukturu i-uzlu z obrázku 1. Jedná se o tzv. *extent-based* souborový systém: i-uzel obsahuje informace o jednotlivých úsecích (extents) souboru ve formě B-stromu. Toto umožňuje nižší režii u extrémně velkých souborů (nejsou-li příliš fragmentovány).

3.4 ReiserFS

Jedná se o souborový systém, který je celý organizovaný ve formě B+ stromu. Byl to vůbec první žurnálovaný souborový systém v Linuxu. Jeho nejzajímavější vlastností je, že alokační jednotkou je jeden bajt, nikoliv jeden sektor. Čili malé soubory zabírají jen nejnutnější místo, bez zarovnávání na velikost sektoru nebo dokonce nějakého většího bloku.

V případě problému prochází `reiserfsck` celé blokové zařízení a hledá na něm signatury uzlů B+ stromu. Což způsobí problémy, pokud například na ReiserFS uložíme soubor, ve kterém je obraz nějakého ReiserFS.

ReiserFS se nicméně již dále nevyvíjí (autoři pracují na vývoji Reiser4) a mimo jiné nepodporuje rozšířené atributy, včetně bezpečnostních kontextů pro SELinux. Nicméně pro netypické zátěže (malé soubory, velké množství souborů v jednom adresáři) je i nadále použitelný.

3.5 JFS

Od firmy IBM pochází souborový systém JFS (Journalled File System), původně implementovaný pro operační systém AIX. Jedná se opět o *extent-based* souborový systém se všemi základními vlastnostmi, které jsou pro Linux potřeba. Některé uživatele možná zarazí, že se žurnál vytváří s velikostí danou pevným podílem z celkové velikosti svazku a tedy na extrémně velkých svazcích může opětovné připojení svazku s JFS (a přehrání transakcí v žurnálu) trvat poměrně dlouho.

3.6 JFFS2

Samostatnou třídou souborových systémů jsou systémy pro solid-state paměti, jako je třeba NAND flash paměť. Základní vlastností flash paměti je, že není třeba (na rozdíl od disků) optimalizovat na sekvenční přístup. Dále pak přepisovatelnost je omezená (průměrná životnost paměťového bloku je 10^4 až 10^6 přepsání). Velikost bloku je větší než u disků (i 32 KB). Na rozdíl od disku, který poskytuje dvě základní operace – čtení bloku a zápis do něj – má flash paměť operace tři: čtení, zápis a vymazání bloku. Před dalším zápisem je třeba blok vymazat. Často mají paměťové bloky k sobě ještě několik bajtů tzv. *out-of-band* dat navíc. Sem lze ukládat další informace o stavu konkrétního bloku.

Těchto zařízení je několik typů: většina USB paměťových karet má v sobě integrovaný paměťový řadič, který interně řeší překlad adres bloků tak, aby bylo zajištěno rovnoměrné opotřebení i při častém zápisu na jedno místo (*wear levelling*). Tyto paměťové karty pak mohou být použity i se souborovým systémem určeným pro disky. Paměťová zařízení která nemají integrovaný řadič s překladem adres (FTL, Flash Translation Layer) vyžadují speciální souborový systém.

Běžné souborové systémy by totiž takovouto paměť častým přepisováním téhož místa (tabulka FAT, tabulka i-uzlů, žurnál apod.) zničily.

Souborové systémy pro flash paměti obvykle nepřepisují datové bloky na místě, ale vytvoří „novou verzi“ datového bloku. Při opětovném připojení svazku je pak třeba zkoumat, který fyzický blok obsahuje nejnovější verzi příslušného datového bloku.

Jedním z aktuálně používaných souborových systémů pro flash paměti je JFFS2 (Journalled Flash File System, verze 2). Jde o žurnálovaný souborový systém se stromovou strukturou, používající mechanismus garbage collection pro opožděnou recyklaci bloků, k nimž existuje novější verze. Jeho nevýhodou je, že při připojení musí projít celou flash paměť, protože metadata souborového systému si drží v paměti RAM počítače. Jak roste kapacita dostupných flash pamětí, začíná tato vlastnost JFFS2 více vadit.

3.7 OCFS2

Souborový systém OCFS2 (Oracle Cluster File System) byl původně vyvinut firmou Oracle pro jejich distribuovaný databázový stroj (RAC, Real Application Cluster). OCFS2 je používán poněkud jinak než ostatní souborové systémy: nepředpokládá se zde výlučný přístup počítače k příslušnému blokovému zařízení. Lze tedy jeden diskový prostor (například diskové pole přes síť SAN – storage area network) připojit jako lokální souborový systém na více počítačích spojených do clusteru. OCFS2 si pak sám řídí komunikaci přes síť i přes toto blokové zařízení, aby pohled více strojů na tentýž svazek byl konzistentní.

Poznamenejme, že pro vyzkoušení OCFS2 nepotřebujeme mít drahé diskové pole a SAN – existují například víceportové disky s rozhranim IEEE 1394, případně můžeme vytvořit distribuovaný systém tolerantní k výpadku pomocí „síťového RAIDu“ zvaného DRBD.[4]

3.8 GFS2

Global File System je projekt zaměřený podobně jako OCFS2, to jest jako souborový systém clusterů se sdíleným diskovým polem. Z projektu GFS2 mimo jiné pochází zamykací nástroj DLM (distributed lock manager), který interně používá nejen GFS2, ale i OCFS2 a další subsystémy.

Příbuzným projektem GFS2 je také CLVM – cluster LVM, tedy nástroj pro koherentní operace nad volume managerem, který pracuje nad blokovým zařízením sdíleným z více počítačů.

[4]Distributed Replicated Block Device, http://www.drbd.org. Pomocí DRBD lze dvě diskové oblasti na dvou různých počítačích zrcadlit a na obou zpřístupnit jako blokové zařízení. S použitím OCFS2 pak může toto zařízení být na obou počítačích i používáno zároveň ve formě souborového systému.

4 Kam směřuje vývoj?

Vývoj v oblasti souborových systémů ale nekončí. V současné době ve světě Linuxu probíhají práce na několika nových souborových systémech. Některé z nich jsou už v relativně použitelném stavu a je možné je testovat. U jiných zatím není jasné, kde až se vývoj zastaví.

4.1 Ext4FS

Souborový systém ext4[5] je následníkem osvědčeného a robustního ext3. Zachovává diskový formát, i když kompatibilní je jen dopředně, nikoli zpětně: svazek ext3 lze připojit jako ext4dev, ale v případě použití nových vlastností ext4dev už není jednoduchá cesta zpět. Výhodou diskového formátu těchto souborových systémů (i včetně UFS, který je na tom podobně) je, že metadata souborového systému jsou víceméně na pevně definovaných místech. Tedy i v případě většího poškození souborového systému je šance na dohledání aspoň nějakých dat. U souborových systémů organizovaných jako B-stromy tuto možnost nemáme – metadata můžou být v podstatě kdekoli a jejich pozice na disku se i v čase mění.

Hlavním rozšířením ext4dev je volitelné zavedení *extent-based* struktury v i-uzlu namísto struktury z obrázku 1. Toto umožní efektivní ukládání velkých málo fragmentovaných souborů a přinese celkové zrychlení pro tyto soubory. Dále implementuje opožděnou alokaci místa pro zápis až v okamžiku zápisu na disk (podobně jako má XFS), časová razítka s rozlišením nanosekund (ext3 a většina dalších souborových systémů má rozlišení sekundové). Odstraňuje limit 32000 podadresářů v jednom adresáři, zavádí kontrolní součet žurnálu pro případ datové chyby.

Poměrně zajímavou vlastností jsou neinicializované *block groups*: v průběhu kontroly svazku programem `e2fsck` se totiž za normálních okolností musí projít tabulky i-uzlů a další struktury v každé BG. Pokud ale nejsou všechny i-uzly využívány, je to zbytečné zdržení. Ext4 si tedy pamatuje, až pokud byla daná struktura uvnitř BG nejdále použita, a tím pádem tato struktura (tabulka i-uzlů, bitmapa volných datových bloků, atd.) nemusí být ani celá inicializována při vytváření svazku programem `mke2fs`, ani celá kontrolována uvnitř `e2fsck`.

Pracuje se i na dalších rozšířeních jako je on-line defragmentace nebo podpora souborů větších než 2 TB.

Souborový systém Ext4 bude dostupný například v distribuci Fedora 9, která bude zveřejněna v květnu 2008.

[5] V současné době již dostupný v oficiálním jádře Linuxu pod jménem ext4dev.

4.2 Reiser4

Následníkem ReiserFS je souborový systém Reiser4. Jeho architektura je velmi podobná jako ReiserFS (včetně alokace místa až na úroveň bajtů), ale přináší některé revoluční vlastnosti. Některé z nich ale způsobují nekompatibilitu s normou POSIX, což vzbuzuje pochybnosti o zařazení Reiser4 do oficiálního jádra.

Základní vlastností Reiser4 je modularita. Souborový systém sám je v podstatě jen varianta B+ stromu na disku s tím, že různé činnosti jsou realizovány pomocí pluginů. Můžeme tak mít například plugin, který pro nově uložené soubory automaticky zajistí jejich indexování vyhledávacím softwarem, plugin pro kompresi a podobně. Soubor může mít více *proudů dat* (stream). Kromě hlavního proudu třeba proud s rozšířenými atributy. Každý soubor je pak vlastně i adresář (svých vlastních proudů), což je právě nepříliš konzistentní s POSIXem.

Reiser4 zpřístupňuje své transakční vlastnosti i pluginům a plánuje se i zpřístupnění aplikacím, takže uživatelský proces bude moci například udělat několik nezávislých operací nad soubory a jejich daty a pak buďto celou tuto sadu změn najednou provést (*commit*) nebo vrátit zpět (*rollback*).

Bohužel vývoj tohoto souborového systému v poslední době příliš nepokračuje[6] a tak není jisté, jestli se vůbec tohoto souborového systému v oficiálním jádře dočkáme.

4.3 BTRFS a CRFS

BTRFS je dalším projektem firmy Oracle v oblasti souborových systémů Linuxu. Jde o souborový systém s kontrolními součty (podobně jako třeba Sun ZFS) a copy-on-write sdílením dat. Používá *extent-based* strukturu i-uzlu, nemá pevně alokovanou tabulku i-uzlů (alokace je až při vytvoření souboru). Je integrován s device mapperem v jádře pro zajištění funkce nad více zařízeními. Umožňuje mít v rámci jednoho svazku více kořenových adresářů (například pro atomické snímky – *snapshots* – svazku v určitém čase). Zajímavou vlastností BTRFS a jeho snímků je, že díky copy-on-write může být kterýkoli snímek i zapisovatelný.

BTRFS je možno začít používat bez reinstalace nad ext3fs – podporuje režim, kdy se ext3 svazek připojí jako BTRFS a teprve postupně dojde k migraci formátu metadat.

Paralelně s BTRFS je vyvíjen projekt CRFS – cache-koherentní síťový souborový systém. Jeho snahou je dosáhnout plně POSIXové sémantiky (na rozdíl od široce používaného NFS) při intentizvním cachování dat na straně klienta. CRFS vyžívá vlastností BTRFS – předpokládá se, že CRFS bude použitý jako server a síťový protokol pro zpřístupňování BTRFS svazků po síti.

[6] Hans Reiser byl v dubnu 2008 odsouzen za vraždu.

4.4 POHMELFS

Alternativním projektem k CRFS je POHMELFS[7] Jevgenije Pojlakova. Podobně jako CRFS je v začátcích svého vývoje, i když je podle všeho o něco dále než CRFS.

Snahou autora je mít síťový souborový systém s více servery a možností odpojeného používání, jen nad lokální cache.

4.5 UBIFS

Pokračováním vývoje v oblasti specializovaných souborových systémů pro solid-state paměti je UBIFS. Používá překladovou (FTL) vrstvu UBI, která již je v jádře v subsystému Memory Technology Devices zahrnuta. UBIFS nad touto vrstvou staví běžný UNIXový souborový systém. Na rozdíl od JFFS2 má strukturu plně uloženou na disku, takže jeho připojení nevyžaduje procházení celé paměti.

Mezi další vlastnosti patří opožděný zápis (write-back), komprese při ukládání dat, žurnálování, možnost synchronních operací. Má dva režimy odpojení svazku: rychlé odpojení, kdy je po připojení potřeba přehrát transakce v žurnálu a normální odpojení, které je pomalejší, ale následné připojení je pak rychlejší, protože seznam neprovedených transakcí je již prázdný.

UBIFS je již téměř ve stavu, kdy je nasaditelný v produčním prostředí.

5 Závěr

Je tedy vidět, že vývojáři souborových systémů v Linuxu jsou stále aktivní a v brzké době můžeme očekávat zajímavé výstupy z několika probíhajících projektů v této oblasti.

[7] Oficiální výklad této zkratky je Parallel Optimized Host Message Exchange Layered File System.

Stabilita souborových systémů

Michal Krátký

E-mail: kratky@fzu.cz

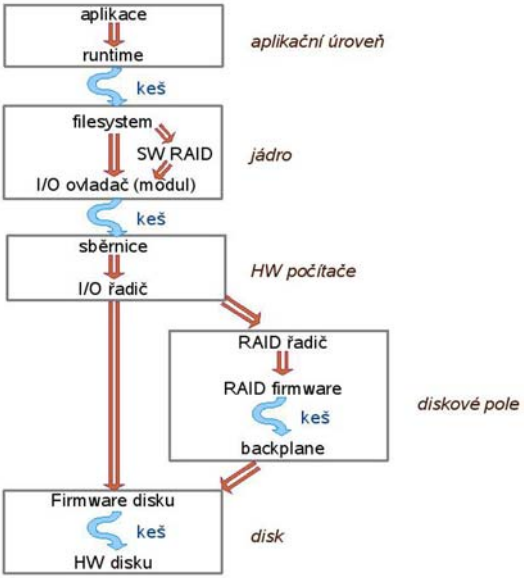
Úvod

V této dekádě narostly zpracovávané objemy dat až na terabyty na jednom počítači a dokonce v jednom souborovém systému (file system, FS). Přitom základní koncepty používané při zpracování dat nedoznaly podstatných změn a pravděpodobnost chyby při jedné datové operaci tedy pravděpodobně nepoklesla. Naopak probíhá adaptace původně serverových konstrukcí do levných komponent a výsledkem je pokles spolehlivosti. Např. SATA disky určené do PC dnes běžně implementují S.M.A.R.T., ale jejich reporty naprosto nelze brát vážně. Výkony levných stolních počítačů jsou nominálně stejné jako výkony skutečných serverů, což vede na jedné straně uživatele (a jejich nadřízené) k „downsizingu“, a na straně druhé výrobce serverů k přechodu na levnější komponenty. Výsledkem je bezpochyby zvýšení pravděpodobnosti poruch při zpracování dat a na to je nutno myslet při vytváření koncepce datových služeb. Ve správě sítě FZÚ jsme opakovaně zažili rozpad FS ext3, který vedl k zastavení provozu, obnově ze zálohy a návratu k bezchybnému provozu bez zásahu do hardware. Tyto zkušenosti krátce popíšeme a v jejich kontextu zhodnotíme výhody a nevýhody tří FS. První je *ext3* – nejběžnější linuxový FS. který jsme napčed používali a potom opustili. Druhý je *xfs* vyvinutý původně firmou SGI a později uvolněný jako Open Source. Pro tento FS jsou typické silné samoopravné mechanismy. Třetí je komerční FS *VxFS* vyvinutý firmou Veritas a nyní prodávaný prostřednictvím fy. SYMANTEC. Tento FS je součástí řady drahých SW produktů, ale ve svojí nejzákladnější podobě je k dispozici zadarmo. Produkty Veritas byly pro FZÚ koupeny, ale dodávka se o pět měsíců opozdila, takže informace zde budou jenom kusé.

Popis problému

Do FS samozřejmě zapisuje výhradně jádro OS. I/O operace je aplikaci potvrzena, ale je jádrem kešována a fyzický zápis proběhne kdykoliv později. Jestliže systém zhavaruje nebo je násilně ukončen, potom při příštím startu může být FS

nekonzistentní, proto musí být před provedením mountu napřed zkontrolován. U žurnálovaných filesystémů tomu tak teoreticky není, neboť transakce jsou zde aktivovány jedinou diskovou operací, která provede přepnutí mezi dvěma konzistentními stavy. Kontrola nečistého FS je před mountem vždy vynucena a kontrola čistého FS před mountem není nutná. Ze všech těchto tak samozřejmých faktů vypývá, že připojený FS je vždy konzistentní.



Obr. 1: Architektura ukládání dat

Ve skutečnosti zápis do filesystému probíhá spíš tak, jak je to naznačeno na obr. 1. Řada operací je kešovaných, ale pořád se zdá, že jedna poslední disková operace „commit“ transakce v žurnálovaném filesystému je pořád jedna disková operace, která se nakonec buď provede nebo neprovede. Disk má možnost zpracovávat více požadavků najednou a při tom si pozměnit jejich pořadí k vůli optimalizaci pohybů hlav. Ale i v tom případě disk potvrdí vyřízení požadavků až po jejich hmotném provedení. Jen když na disku povolíme režim *write back*, bude disk potvrzovat provedení diskových operací ihned po jejich vložení do keše a jádro potom ztratí kontrolu nad jejich skutečným provedením. Jestliže potom takový systém násilně vypneme, zůstanou některé diskové operace neprovedené a souborový systém může být nekonzistentní, i když je žurnálovaný. Toto nebezpečí je reálné, ale zřejmě není jediné, neboť jsme pozorovali rozpady souborových systémů i bez násilného vypnutí. Samozřejmě to znamená, že něco nefungovalo jak má.

Poznámka: *Write back* lze nastavit i na diskových polích a znamená to potom, že poli natolik důvěřujeme, že mu svěřujeme samostatné provádění velmi rozsáhlých souborů datových operací. Často je potřeba tuto funkcionalitu povolit, protože kešování je hlavní prostředek zvyšování výkonu pole a zrychlení bývá mnohonásobné.

Zkoumané rozpady souborových systémů byly pozorovány za podmínek občas chybujícího disku, dále na serveru, který byl velmi zatížen diskovými operacemi nad laciným diskovým polem, ale také za běžného provozu výpočetního uzlu. Případ osobního počítače s chybujícím diskem naznačuje možnost, že porucha v datech vznikla při automatickém přiřazení náhradních bloků. Disk se očividně opravil, ale data byla narušena. Případ s přetíženým serverem a low end diskovým polem může naznačovat chybu firmware pole. Případ rozpadu výpočetního serveru s normálním provozem nenaznačuje nic. Stroj od té doby běží další dva roky bez jakékoliv změny HW nebo SW. Popsané úkazy proběhly pod linuxem na FS ext3. Tento filesystém je zřejmě nestabilní v tom smyslu, že porucha ve strukturních datech se za provozu šíří, až zničí celou adresářovou strukturu. Po přechodu na FS *xfs* se podobné projevy zatím nevyskytly. Jestliže je správná domněnka, že xfs je ve výše uvedeném smyslu stabilní, potom je také pravděpodobné, že v datech na discích existují poruchy, které se nešíří a nikdo o nich neví.

Hledání nekonzistencí

Jakmile jsme pojali toto podezření, budeme si nejspíš přát předpokládané poruchy najít, ideálně za chodu systému. Nabízí se myšlenka udržovat někde stranou tabulku kontrolních sum existujících souborů. Kontrola by potom spočívala především v testu, jestli u souboru, jehož obsah se podle řídících údajů inode neměl změnit, se kontrolní suma také nezměnila. Taková kontrola by ale našla jen takové poruchy, které vznikly ex post porušením původně správného obsahu na disku. Zřetelně lepší by bylo integrovat kontrolní sumy přímo do mechanismu FS. Toto snad dělá SUN *zfs* a je možné, že pouhým jeho použitím bude celý problém vyřešen.

Bez kontrolních sum je hledání chyb v obsahu dat vyloučeno. Můžeme se snažit hledat aspoň chyby ve strukturních datech, to znamená nekonzistence FS. Pouhé prolezení celé adresářové struktury a přečtení obsahu všech datových souborů je určitý test konzistence. Každá úplná záloha je proto dobrá, i kdyby měla jít na zařízení /dev/null.

Také jádro OS může za běžného provozu narazit na nekonzistenci souborového systému a je ovšem otázka, jak se v tom případě zachová. V linuxu pro FS *ext2/ext3* existuje volba

errors=continue | remount_ro | panic

kterou lze použít při mountu nebo nastavit jako defaultní chování FS pomocí utility *tune2fs*. Hodnota *continue* znamená, že systém má i při výskytu I/O chyby pokud možno pokračovat v činnosti, *remount_ro* zakáže zápisy do problematického FS a *panic* rovnou vyvolá paniku. (Zdali bude systém rebootován nebo zůstane dole závisí na parametrech bootu.) Volba *errors=* se uplatní obecně v případě I/O chyb.

Podobně pro *vxfs* má mount volbu

ioerror=disable | nodisable | wdisable | mwdisable

kde *disable* znamená okamžité odpojení FS, *wdisable* znamená znemožnění zápisu, *mwdisable* znamená znemožnění změn ve strukturních datech. *Nodisable* znamená pokračování v práci, ale inody postižené chybou jsou pro další práci zablokovány. *xfs* nic podobného neumí.

Seriózní vyhledávání nekonzistencí je samořejmě možné jen při odpojeném FS. Z výše uvedených úvah vychází, že je potřeba kontrolovat i FS označený jako „čistý“. Servery jsou obvykle provozovány nepřetržitě. Preventivní odstávky systému na profylaktickou údržbu už nejsou v módě a dodavatelé i provozovatelé se často dokonce chlubí, že mají uptime stovky dní, aniž by jim systém spadnul. Když už jednou za čas dojde k odstávce, je tím spíš vhodné věnovat trochu strojového času aspoň základní údržbě. Pro *ext2/ext3* tedy napíšeme

fsck -f -y *zařízení*

Jestliže máme podezření na chybující disk, můžeme v případě FS *ext2/ext3* použít povel

fsck -cck -y *zařízení*

V tom případě program fsck před kontrolou konzistence napřed otestuje všechny bloky zadané diskové oblasti na čtení a zápis pomocí programu *badblocks*, a pokud se najdou chybující bloky, zapíše si je do speciálního inode určeného právě pro vadné bloky. Tím jsou tyto bloky interně zarezervovány, čímž je vyloučeno jejich použití pro data. Tento mechanismus nemá nic společného se seznamem defektů, který je udržován firmwarem disku. Může se také stát, že v průběhu testování povrchu se blok projeví jako chybující, jako takový je ve FS označen, a současně je opravným algoritmem disku zaměněn za jiný blok z náhradního prostoru, takže oprava ve FS byla vlastně zbytečná. Takové případy nelze zjistit, protože seznam defektů disku je zvenčí neviditelný.

Pro *vxfs* zadáme úplnou kontrolu povelem

fsck -y -o full *zařízení*

FS *xfs* má jen utility **xfs_check** a **xfs_repair**, které neumožňují vynutit úplnou kontrolu. Jeho autoři byli zřejmě toho názoru, že kvatliní software je samospasitelný a chyba v datech není možná.

Zabezpečení redundancí

Pro zlepšení zabezpečení uložených dat obvykle používáme redundanci. Lze to udělat na různých úrovních. Některé aplikace jsou napsány tak, že umějí pracovat nad dvěma případně i více exempláři dat. Pouze jeden exemplář je v každé chvíli aktivní, ostatní jsou záložní. Je-li indikována chyba dat, má administrátor možnost přepnout na záložní médium. Pokud chyba vznikla jednorázovým selháním diskového subsystému, bude tímto napravena. Indikovat chybu je ovšem nutné uživatelsky na aplikační úrovni.

Je možné, aby přímo v souborovém systému byla zavedena redundance, žádný z posuzovaných FS to ale nepodporuje. Zavedení redundance na kterékoliv vyšší úrovni, než kde je očekáván výskyt chyby (SW RAID, HW RAID), v naší situaci může pomoci, přestože prostředky redundance jsou tradičně stavěny na problém nečitelného bloku nebo nefunkčního disku. Např. když máme SW RAID a pod ním je porušený obsah dat jednoho diskového bloku, potom kontrola integrity RAIDu musí chybu odhalit, ale není ovšem jak ji opravit, protože není známo, která verze dat je správná.

Poznámka 1: Dělat pravidelné kontroly integrity RAIDu je vhodné také proto, že při ní jsou disky v poli hodně namáhány. Jestliže je některý disk už na rozpadnutí, dostane příležitost se projevit. Provozovat diskové pole, u něhož se bojíme, aby nám neselhalo více disků najednou, je zcela nevhodné. Na rebuild stejně jednoho dne dojde, a o data pravděpodobně přijdeme.
Poznámka 2: Můžeme si uvědomit, že redundance je něco jiného než záloha. Když snaživý administrátor zařídí pravidelné zrcadlení FS na jiný FS programem *rdist* nebo podobným, jedná se o zálohu. Data napřed prošla všemi vrstvami až na disky, a potom stejnou cestou procházejí nazpátek na aplikační úroveň a odtud na záložní médium. Záloha samozřejmě chrání před všemi druhy výpadků, ale je to za cenu návratu ke staršímu časovému bodu.

Závěr

Souborový systém *ext2/ext3* hodnotíme jako nestabilní, protože nekonzistence se v něm lavinovitě šíří. Souborový systém *xfs* se v provozu osvědčuje velmi dobře, ale je rizikový, protože se plně spoléhá na automatické kontroly a opravy a nedává možnost ručního zásahu do procesu opravy dat. Souborový systém *VxFS* je navržen jako velmi robustní a dává i slibné prostředky pro záchranu poškozených dat. Praktické provozní zkušenosti ale ještě nemáme. Zabýváme se situací, kdy diskový subsystém velmi zřídka chybuje. Žádný z posuzovaných souborových systémů nenabízí rozumnou metodu, jak vzniklé chyby v datech za běhu systému identifikovat nebo eliminovat. Nezbývá než počkat si na další vývoj v oblasti ukládání dat.

Volume management a Linux

Milan Brož

E-mail: mbroz@redhat.com

Abstrakt

Popis specifických částí LVM2, jaderného ovladače device-mapper, jak se liší od MD (multiple device). Příklady nástrojů využívajících device-mapper, cryptsetup a LUKS. LVM2 – základní vlastnosti, metadata.

1 Úvod

Základní myšlenky správy diskového prostoru jsou v různé podobě součástí většiny operačních systémů.

Základními požadavky na správu diskového prostoru jsou zejména:

- abstrakce nad omezení jednotlivých typů fyzických zařízení
- jednotné ovládání, flexibilita a škálovatelnost
- operace za běhu (High Availability)
- zvýšení bezpečnosti a spolehlivosti (RAID, snímky obsahu)

Operační systém Linux jako velmi flexibilní systém by měl poskytovat adekvátní nástroje pro správu diskového prostoru a výběr souborového systému.

Následující popis je popisem vrstvené architektury, která má podporu v současném linuxovém jádře. Srovnání použití alternativních systémů pro konkrétní konfiguraci je již úkol pro laskavého čtenáře.

Pojmem Volume management je zde chápán systém správy diskového prostoru na úrovni blokového zařízení, tedy vrstvy pod vlastním souborových systémem.

Toto řešení však nijak nevylučuje kombinace s jinými implementacemi volume managementu či souborového systému.

V tomto textu si krátce popíšeme, co lze očekávat od LVM2 a ostatních aplikací využívajících device-mapperu (což je specifická funkce linuxového jádra).

2 MD, LVM, device-mapper, ... co je to?

Celá architektura LVM2 [1] je postavena na rozdělení funkční části na jaderný ovladač, nazývaný **device-mapper**, který implementuje základní funkce pro práci s blokovými zařízeními a na **část implementovanou v userspace**, které pomocí rozhraní device-mapperu zajišťuje všechny funkce spojené s volume managementem.

2.1 Jaderné ovladače – device-mapper

Vlastní device-mapper je modulární a velmi flexibilní nástroj, který jednoduchým způsobem umožňuje implementovat prakticky libovolnou kombinaci mapování blokových zařízení, včetně následných rekonfigurací, to vše za běhu systému a přítomnosti aktivních IO operací nad rekonfigurovaným zařízením.

LVM2 tedy vždy vyžaduje jádro s funkčním device-mapperem.

Vlastní použití device-mapperu není omezeno jen na LVM2 (ve skutečnosti LVM2 zdaleka nevyužívá veškeré možnosti), proto si v další části popíšeme device-mapper podrobněji.

Ačkoliv se od administrátorů neočekává, že budou pracovat s rozhraním device-mapperu přímo (na to lépe slouží nástroje jako *LVM2 tools*, *dmraid*, *cryptsetup*), je dobré o této vrstvě systému vědět více.

2.2 MD – multiple devices

MD je historicky nejstarší a velmi stabilní subsystém, který umožňuje vytvářet nové blokové zařízení která spojí existujíci disky do RAID mapování. Víceméně se dnes používá RAID 0,1,10,5 a 6.

Většina funkcí je implementována přímo v jádře, a ovládá se přes utilitu *mdadm* [12] (která pokrývá také všechny funkce původních raidtools).

Z principu funkce device-mapperu je zřejmé, že všechny funkce MD by šly implementovat pomocí device-mapperu. Některé funkce jsou opravdu v jádru možné implementovat obojími nástroji, ale protože se použití nijak nevylučuje, častějším řešením je spíše kombinace obou nástrojů (například LVM2 které používá MD RAID5 zažízení).

A proč nejde použít funkce MD přímo? Problémy spočívají právě v návrhu MD jako specifického jednoúčelového nástroje – chybí zde funkce, poskytující základní stavební prvky pro integraci v device-mapperu (změna mapovacích tabulek zařízení, některé funkce jsou implementovány přímo v jádře, u device-mapperu je to očekáváno v userspace apod.).

Nejkomplikovanějším problémem je však použití v clusteru – DM a LVM2 bylo od začátku navrhované pro možné použití na sdíleném diskovém prostoru

(tj. více strojů vidí stejný diskový prostor), kdežto MD je určeno výhradně pro lokální zažízení.

3 DM – device-mapper

Device-mapper je jaderným ovladačem umožňujícím vytvářet nová bloková zařízení a mapovat je na existující (včetně jiných zařízení vytvořených device mapperem – tzv. device stacking).

Device-mapper ve skutečnosti neví nic o pojmech volume managementu, ani o dělení disku na particie, ani o použitých filesystémech na těchto zařízeních. Základním stavebním kamenem je *blokové zařízení* a adresovatelnou jednotkou je *sektor*.

Ovládání probíhá pomocí několika *ioctl* příkazů, pro uživatelské aplikace existuje zastřešující knihovna *libdevmapper* [2] (s čestnou výjimkou EVMS [11], který s device-mapperem komunikoval přímo).

Volání device-mapperu na nejnižší úrovni uživatelem umožňuje příkaz *dmsetup*.

Základní funkce spočívají v nastavení mapovací tabulky zařízení, která specifikuje jméno nového zařízení a popis jak jsou jednotlivé sektory dále mapovány. Tabulka zařízení se může měnit za běhu systému, device-mapper poskytuje funkce pro zmrazení aktivních operací a výměnu mapovací tabulky. Zařízení může mít dvě mapovací tabulky, *aktivní* a *neaktivní*. Jedním atomickým krokem lze pak neaktivní mapovací tabulku přepnout do aktivního stavu. (Obecně lze tedy změnit mapování aniž by běžící aplikace něco poznala.)

Na následujících příkladech lze jednoduše ukázat, co to je mapovací tabulka (příklady jsou vybrány spíše pro jednoduchost, na této úrovni administrátor normálně zařízení nevytváří.)

Podobné postupy však lze použít například v regresních testech pro simulaci zařízení a stavů, které nejsou obvyklé.

Jiným praktickým použitím je Live DVD, kde je na read-only disku uložen vlastní souborový systém a uživateli se v paměti (ramdisku) vytvoří pomocí device-mapperu zapisovatelný snímek, uživatel vidí tedy zapisovatelný kořenový souborový systém aniž by reálně používal disky vlasního stroje.

Příklad – simulace vadného sektoru na disku

```
dmsetup create bad_disk <<EOF
0 8     linear /dev/sdb1 0
8 1     error
9 10000 linear /dev/sdb1 9
EOF
```

Příklad – 8TB disk, kde je však zapisovatelných pouze prvních 32k

```
dmsetup create fake_disk  <<EOF
 0          64 linear /dev/sdb1 0
64 17179869120 zero
EOF
```

Příklad – šifrovaný disk (bez použití cryptsetup)

```
dmsetup --table "0 $SECS crypt aes-cbc-essiv:sha256 $KEY 0 $DEV 0"
```

Nově vytvořená zařízení jsou dostupná v adresáři */dev/mapper*.

Popis jednotlivých mapovacích tabulek je součástí dokumentace linuxového kernelu a v některých dalších článcích [10].

Targety

Vlastní rozsah funkcí device-mapperu je dán dostupnými *targety*.

Target je plugin, tedy nezávislý jaderný modul, který implementuje konkrétní typ mapování. V mapovací tabulce lze pak jednotlivé targety kombinovat.

- **linear** – mapuje jednoduchý rozsah bloků na jiné zařízení
- **striped** – odpovídá RAID0 (striping)
- **mirror** – odpovídá RAID1 (mirror) je základním prvkem pro online přesun (pvmove) Pro tyto přesuny je implementován kernelový démon kcopyd, který asynchronně provádí zapisy na blokové zařizení.
- **crypt** – šifruje blokové zařízení s použitím cryptoAPI
- **snapshot** – umožňuje vytvářet virtuální snímky zařízení v čase
- **zero, error, delay** – pomocné targety
- **multipath** – umožňuje správu multipath zařízení (tedy blokových zařízení, ke kterým má systém více aktivních cest, například přes Fibre Channel switch) Konfigurace vyžaduje speciální uživatelské nástroje. Z pohledu volume managementu je vidět opět jedno blokové zařízení, multipath nástroje tedy jen konfigurují aktivní cestu k danému uložišti a vybírají patřičnou strategii přístupu, výběru cesty a load balancingu.)

Vzhledem k aktivnímu vývoji existují další dostupné moduly, ty jsou ale zatím mimo stabilní jaderný strom.

- **raid45** – RAID4,5 implementace pro dmraid

- **loop** – alternativní implementace pro loopback
- **ioband** – přidělování přenosové kapacity jednotlivých blokových zařízení podle definovaných skupin procesů, například pro několik virtuálních strojů sdílejících jeden fyzický disk.
- **replicator** – implementace vzdálené replikace zařízení

3.1 DMRAID

DMRAID je nástroj sloužící pro zpřístupnění diskových polí vytvořených pomocí software RAID řadičů. Tyto řadiče bývají někdy označovány jako ATARAID [5]. Jde o levné řadiče, často s vlasním BIOSem, který umožňuje vytvářet RAID pole. Vlastí funkce je však podmíněna ovladačem, který implementuje vlastní RAID algoritmus s pomocí systémového procesoru.

Funkce DMRAIDu je velmi prostá – proprietární metadata, uložená při konfiguraci RAID pole na disky, převede na mapovací tabulku device-mapperu a zpřístupní tak dané diskové pole systému.

Vzledem k tomu, že jde o softwarový RAID, tato konfigurace má zejména smysl při sdílení tohoto pole s jiným operačním systémem. Pro čistě linuxové instalace mnohem lépe poslouží MD RAID, případně přímo LVM2.

3.2 cryptsetup-LUKS

Šifrování dat je v poslední době vyžadováno z různých důvodů. Může jít o požadavek ochrany dat v notebooku, ale i ochrana před neoprávněnou manipulací s daty na serveru, případně podmínka pro certifikaci systému pro manipulaci s citlivými daty.

Šifrování dat je možné na různých úrovních – na úrovní souborového systému (například EncFS [6]), případně na úrovni blokového zařízení s pomocí dm-cryptu [7].

V případě šifrovaného blokového zařízení je šifrováno opravdu vše, včetně metadat daného souborového systému. Toto řešení je jako jediné použitelné pro šifrování swapu.

Na druhou stranu, šifrování na úrovní filesystému umožňuje selektivní výběr souborů, a zároveň umožňuje kopírovat data v jejich šifrované podobě.

DM-crypt umožňuje využívat všechny šifrovací algoritmy implementované v rámci jaderného cryptoAPI. Od verze jádra 2.6.25 také přímo podporuje asynchronní cryptoAPI, což dovoluje využít hardwarové prostředky (jako jsou specializované cryptoprocesory). Tím nejen v budoucnu umožní odlehčit systémovému procesoru, ale zároveň umožní použít certifikovaný hardware.

LUKS (Linux Unified Key Setup)

Device-mapper poskytuje základní prostředky pro vytvoření šifrovaného disku, ale podstatnou částí řešení je také správa klíčů a zajištění přenositelnosti takto šifrovaného zařízení mezi jednotlivými distribucemi, případně i do jiného operačního systému.

Zde přichází na řadu LUKS [8]. LUKS zajišťuje standardní formát pro šifrovaný disk. Tento formát nejen že zajišťuje možnost přenositelnosti disku mezi systémy, ale zároveň přidává některé další vlastnosti zvyšující bezpečnost uložených dat.

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

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

LUKS hlavička, obsahující metadata pro správu klíčů a parametry použité pro šifrované zařízení je uložena na počátku disku, obdobně jako metadata pro LVM2.

Práce s LUKS diskem

vytvoření LUKS a klíče (s defaultními parametry)

```
cryptsetup luksFormat $DEV
```

zpřístupnění šifrovaného zařízení

```
cryptsetup luksOpen $DEV crypt1
```

Případně zrušení mapování

```
cryptsetup luksClose crypt1
```

Samozřejmě je možné takto vytvořené zařízení používat v kombinaci s LVM2. Je možné šifrovat celý disk, nebo jen některé oddíly. Většina distribucí již podporuje nastavení šifrování přímo v */etc/cryptab*, ale například Fedora 9 už podporuje přímo instalaci na plně šifrovaný kořenový svazek (včetně šifrovaného swapu).

4 LVM2 – Linux Volume Management

LVM1 bylo dostupné již pro jádro řady 2.4 ale implementace byla odlišná (implementace některých částí přímo v jádře, použití jiného, binárního, formátu metadat). Z dnešního pohledu je LVM1 již zastaralé.

Přesto LVM2 nástroje podporují všechy základní příkazy LVM1 a umí pracovat s LVM1 formátem metadat.

LVM2 vzniklo jako obecný návrh, vycházející z rozdělení funkcionality mezi jádro systému a uživatelské nástroje. Zároveň byl hned od počátku kladen důraz na použití v rámci clusteru (vývoj LVM jde paralelně s vývojem GFS).

4.1 Architektura PV,VG,LV

Z pohledu uživatele je architektura na první pohled komplikovaná, ale po pochopení základních priincipů je ovládání vcelku jednoduché.

Logicky lze LVM2 rozdělit do 3 úrovní:

- **PV (Physical Volume)** – odpovídá logicky fyzickému disku, který je spravován LVM2. Vytvoření PV je ve skutečnosti jen zapsání jednoduché hlavičky, která identifikuje zařízení jako součást LVM2 a vyhradí na počátku disku místo pro LVM2 metadata.

- **VG (Volume Group)** – je skupina disků, ze kterých se přiděluje místo pro logické oddíly. (Tvoří tak vlastně jakýsi pool pro jednotlivé logické oddíly).

- **LV (Logical Volume)** – jsou pak logické diskové oddíly, které již uživatel používá jako disky.

4.2 Metadata

Popis kongfigurace je uložen v texovém formatu (metadata), který je uložen ve vyhrazené části na začátku disku. Standarně jsou metadata redundantně uložena na všech PV patřících do VG. Metadata jsou uložena v kruhovém bufferu (existuje tedy, podle velikosti, i několik předchozích kopií) a zároveň je udržován kontrolní součet a sériové číslo metadat.

Zároveň se metadata při každé změně automaticky zálohují do souboru (standardně v */etc/lvm/backup*).

Existují tedy předpoklady pro poměrně jednoduchou opravu poškozených metadat v případě havárie. Současné nástroje sice poskytují nástroje pro takovou manipulaci a obnovu metadat, nikoliv však ještě patřičnou heuristiku pro odstranění případných kolizí.

(Ale i v komplikovaných případech existuje cesta, kdy zkušený administrátor může obnovit poškozená data, v extrémním případě lze ruční editací metadat zachránit prakticky cokoliv.)

4.3 Příkazy

Cílem tohoto článku není podrobně popsat příkazy LVM2 – čtenáře lze odkázat na velmi hezky zpracovanou dokumentaci, která je součástí Red Hat Enterprise Linuxu [3].

Pro uživatele, kteří znají VxVM (Veritas Volume Manager) může přijít vhod stručné srovnání příkazů s LVM2 [4].

Příkazy obecně používají předponu *pv/vg/lv* podle toho, se kterou entitou se pracuje. Stručný popis lze získat zadáním **lvm help** a dále popisem v manuálové stránce. Reportovací přípazy (pvs/vgs/lvs) umožňují vypsat všechny atributy v potřebném formátu pro použití ve skriptech, stručný popis lze získat zadáním atributu help (tedy například **pvs -o help**).

4.4 Příklad

Na následujícím přikladu lze demonstrovat vytvoření VG, obsahujícího dva disky a vytvoření jednoho RAID0 logického oddílu.

```
fdisk /dev/sd[bc]
pvcreate /dev/sd[bc]1
vgcreate vg_test /dev/sd[bc]1
lvcreate -L 400M -i 2 -n lv_strip vg_test
...
mke2fs /dev/vg_test/lv_strip
mount /dev/vg_test/lv_strip /mnt/tst
...
umount /mnt/tst
...
lvremove vg_test/lv_strip
vgremove vg_test
pvremove /dev/sd[bc]1
```

4.5 Podporované funkce

LVM2 poporuje následující základní vlastnosti (samozřejmě v kombinacích s MD RAIDem, cryptsetupem a podobnými nástroji lze vytvořit další varianty).

- Vytváření, online změna parametrů a velikosti logických oddílů

- Online přesun libovoného oblasti na PV
- RAID0 (striping), RAID1(mirroring) a manipulace s nimi
- Snímky obsahu
  - zapisovatelné snímky na úrovni blokového zařízení nezávislé na souborovém systému
  - používá copy-on-write oddíl, vyžaduje tedy předem vyhradit místo na VG
- fsadm – nástroj pro správu LV a souborového systému

4.6 Cluster LVM

Podpora funkcí LVM2 v prosředí clusteru (tj. skupině systémů, které sdílejí blokové zařízení) vyžaduje clvmd démona, který zajišťuje distribuované zamykání (na úrovni VG a LV) s pomocí clusterové infrastruktury.

S výjimkou snapshotů jsou v clusteru v aktuální verzi podporovány všechny základní vlastnosti LVM2, ovládání je zcela identické s lokálními příkazy.

Podmínkou pro použití mirroru a pvmove (s aktivními oddíly) je přítomnost cluster log modulu a démona (součást Red Hat Cluster Suite).

4.7 Náročnost na systémové prosředky

Častou otázkou bývá jak používání LVM (device-mapperu) ovlivňuje výkonnost ve srovnání s přímým přístupem na fyzické zařízení.

Většina problémů je však svázána s konkrétní konfigurací, obecně při mapování jednoduchých logických oddílů je ztráta výkonu zanedbatelná.

- Při vrstvené konfiguraci (LVM nad MD) a nevhodném nastavení může dojít k tomu, že jednotlivé zřízení nad sebou mají odlišný počátek bloku (chunku) a při přístupu zbytečně generují další IO operace (RAID chunk, souborový sytém a počátek LV - metadata area)
- Nastavení readahead – některé konfigurace (RAID0) vyžadují pro dosažení stejného výkonu při sekvenčních operacích vyšší readahead. V nových verzích LVM2 je optimální readahead nastaven automaticky.
- Pokud se používá mnoho disků a logických oddílů (řádově desítky), které obsahující LVM metadata, některé LVM operace mohou trvat poměrně dlouho. V tomto případě se doporučuje neukládat metadata na všechny disky (automatická správa metadat při velkém počtu PV bude součástí další verze LVM2).

5 Závěr

V poslední době se objevilo několik rozličných diskuzí, jak by měl volume management vypadat.

Názory se liší od striktního oddělení jednotlivých vrstev (blokové zařízení, RAID, souborový systém) až po jednotný nástroj, který spojuje všechny vlastnosti dohromady.

Zároveň se zcela logicky objevují hlasy, zda je žádoucí, aby některé funkce byly v jádře Linuxu implementovány duplicitně.

Každý přístup má své výhody, u každého se najde situace, která je pro dané použití neefektivní, složitá, nebo případně nejde použít vůbec. Nemyslím, že v případě Linuxu existuje jedno perfektní řešení.

Z open source nástrojů je zde vhodné ještě zmínit existenci EVMS (Enterprise Volume Management System), který měl poměrně rozsáhlé možnosti, jeho vývoj je ale na mrtvém bodě a jeho podpora byla víceméně ukončena.

LVM2 a device-mapper existuje již delší dobu, účel a implementace se ukázala jako dobře životaschopná a jsou stále přidávány nové funkce (ač možná pomaleji, než by dychtiví uživatelé očekávali).

Prioritou vývoje device-mapperu je co největší integrace do blokové vrstvy tak, aby byl kód co nejvíce transparentní a nekomplikovaný. Zároveň probíhá vývoj nových funkčních vlastností. (Například škálovatelná implementace snímků, replikace logických oddílů, apod.) Některé funkce narážejí pouze na absenci kódu v uživatelských nástrojích – jádro potřebné nástroje již obsahuje (hotspare management, kombinace snímků a RAIDu, sledování statistik IO operací).

Literatura

[1] http://sources.redhat.com/lvm2/

[2] http://sources.redhat.com/dm/

[3] http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Cluster_Logical_Volume_Manager/index.html

[4] http://people.redhat.com/sfolkwil/lvm_vxvm.txt

[5] http://linux-ata.org/faq-sata-raid.html

[6] http://www.arg0.net/encfs

[7] http://www.saout.de/misc/dm-crypt/

[8] http://luks.endorphin.org/

[9] http://people.redhat.com/agk/talks/

[10] http://mbroz.fedorapeople.org/talks/

[11] http://evms.sourceforge.net/

[12] http://www.kernel.org/pub/linux/utils/raid/mdadm/

ZFS — The Last word in Filesystems

Vineeth Pillai

E-mail: vineeth.pillai@sun.com

Abstract

ZFS is a new revolutionary filesystem developed in SUN Microsystems that has fundamentally changed the way filesystems are administered. A completely open source project which throws away the ideas of conventional filesystem and designed from scratch to be robust, scalable and simple to administer. This paper aims to be an introductory material for developers and administrators to be able to understand ZFS and appreciate the features and advantages of this powerful feature rich yet simple to administer filesystem.

1 ZFS Overview

ZFS is a transactional object based filesystem. It is a 128bit filesystem and hence have virtually unlimited storage capacity based on the underlying disk that you have and is not limited by the software.

ZFS uses the concept of pooled storage. Conventional filesystems were created on top of a single physical device. To accommodate multiple devices, the concept of volume manager was brought in and this added another layer of complexity. Filesystem had no control over the physical devices and it had to rely on the stability of volume manager. ZFS eliminates the need for volume management by aggregating all the available physical devices into a storage pool. This pool acts as an arbitrary data store from which filesystems could be created. Filesystems no longer needs to worry about the size constraints as long as you have enough storage backup at the devices level. New devices can be seamlessly added to the pool and all filesystems with in the pool can immediately access the additional space without any additional work.

ZFS ensures that the data would be always consistent on the disk. This avoids the need for additional programs to check the consistency of filesystem.

More over all data and metadata are checksummed using a user-selectable algorithm. All checksumming and data recovery is done at the filesystem layer and is transparent to the applications.

Self healing property of the data is another great feature of this filesystem. ZFS supports storage pools with varying levels of redundancy. ZFS recognizes bad data blocks and when a bad data block is detected, it fetches the correct data from a redundant copy and then corrects the bad data.

2 History: Why ZFS?

Existing filesystems had many drawbacks which pushed the need for coming up with a new design for filesystems.

a) Silent Data corruption is one of the main problems faced by existing filesystems. A defect in the disk controller, driver or firmware can corrupt the data silently without the filesystem coming to know about it.

b) Traditional filesystems are very difficult to manage. User needs to have knowledge of the filesystem details like labels, partitions, volumes etc. It becomes very difficult when it comes to growing or shrinking an already existing filesystem.

c) User has to depend on various set of tools and programs to manage filesystems.

d) Portability is also an issue for the traditional filesystems.

e) When it comes to performance, there are many bottlenecks in the traditional design like fat locks, fixed block size, naīve prefetch, growing backup time etc which makes the filesystems very slow.

3 ZFS: Objectives

While ZFS was designed, all the current flaws of traditional filesystems were kept in mind and 20 years of obsolete assumptions were blown away. It was figured out as to why storage has gotten so complicated and designed an integrated system from scratch. To name a few design objectives:

a) Filesystem should be easy to administer.

b) Filesystem should do the work for you.

c) Data should be consistent on the disk always.

d) Filesystem should be architecture independent. It should be portable.

e) All the above features without compromising on the performance.

4 ZFS Features

4.1 ZFS Pooled storage

ZFS has the concept of pooled storage where in all the physical devices are abstracted as a storage pool. Filesystems are created on top of this pool and user doesn't have the pains to allocate resources for a filesystem. ZFS manages everything for the user. Disks can be added seamlessly to the pool and filesystem transparently gets to use the additional disk space. Advanced options are also available where user can specify quotas and reservations on the filesystems. The best analogy that can be quoted is 'ZFS works on disks as VM(virtual memory subsystem) works on memory'. When we add or remove a memory module, OS handles the work and everything is transparent to the user. Similarly ZFS handles the disks or storage for the user.

4.2 Data Integrity and Security

Integrity of the data on the disk is the most essential feature that is demanded of any filesystem. Most conventional filesystem overwrite blocks when modifying in-use data. In case of a power failure during these operations, the data is corrupted and filesystem may lose block pointers to important data. There are add-on utilities to filesystems which scans the corrupted filesystems and tries to correct it. This action might take long time for completion and are not guaranteed for successful recovery.

ZFS uses a concept called copy-on-write. Blocks containing the in use data are never modified. The changed information is written to alternate blocks and the block pointer to the in use data is only moved once the write transactions are complete. If there is a power outage in between the data write, no corruption occurs the pointer to the good data is not altered until the entire write is complete. This avoids the need of add-on utilities to keep track of the health of the filesystem which is really expensive.

4.3 End-to-end Checksumming

Most filesystems only protect against bitrot as the checksum is stored along with the data block. This won't protect the data from issues like:

- Phantom writes where driver returns write but actual write didn't take place due to controller error or driver error

- Misdirected reads or writes where the disk accesses the wrong block.
- Accidental overwrites.
- DMA parity errors

In ZFS, the checksum is stored in the pointer to the block rather than with in the block, all the way up to the root block also called the uberblock. All block checksums are done in the server memory, so any error up the tree is caught directly. In a mirrored configuration, along with the detecting the problem, zfs also corrects the problem by replacing the corrupt data with the correct data from the good mirror.

ZFS has a low priority process which constantly checks for any inconsistencies in the filesystem. This is called scrubbing. Since this is of very low priority, it won't affect the important programs running on the server.

4.4 ZFS Scalability

ZFS is a 128bit object based filesystem which can achieve unlimited storage theoretically. ZFS also implements data pipelining, dynamic block sizing, intelligent prefetch, dynamic striping, and built-in compression to improve performance.

In addition to the 128bit capability, ZFS metadata is purely dynamic. As a result creating a storage pool and filesystems are extremely fast. Only one to two percentage of writes to disks are metadata. As there is no static metadata, there is no restriction in the number of files and it depends on the disk capacity only. You can have built-in encryption also, if your data needs to be secured.

4.5 ZFS snapshots

This is one of the desirable features of a filesystem where you are able to take the snapshot at any particular point in time and be able to revert back if needed. ZFS handles this seamlessly with the copy-on-write concept. The time complexity is O(1) compared to O(N) in traditional filesystems. Snapshots start to occupy disk space only when the active dataset changes and the additional disk space consumed is just the difference between snapshot and active data set.

4.6 RAID-Z

RAID-Z is a variant of RAID5 built on top the ZFS concepts. It has dynamic stripe width and variable block size ranging from 512bytes to 128KB. All writes are full stripe writes and it eliminates the RAID-5 write home and there is no need for NVRAM. RAID-Z doesn't have a requirement for special hardware and it is most comfortable with cheap disks.

4.7 Ditto Blocks and built-in compression

ZFS maintains data redundancy above and beyond mirrors and RAID-Z. Each logical block of data can have up to three physical replica blocks in the disks. Metadata is also replicated the same way on the disk. These data are compressed and stored on disk through which we can afford to have multiple copies on the disk. This ensures the availability of data even if we have corrupted data on the primary block.

5 ZFS Administration

With pooled storage concept, user no longer need to worry about the volumes. You can have upto 2^{48} datasets per pool. ZFS provides you the ability to administer such a power packed, feature rich filesystem with the help of just two main simple commands. zpool(1M) command is used for managing and administering the pools and zfs(1M) is to configure, manage and administer ZFS filesystems.

Filesystems are the administrative control points and they are hierarchical by design. Properties are inherited from the parent and users can change the properties of individual datasets. Administration can be delegated to normal users where by users are made capable of managing their own data.

To detail all the commands and options is out of scope in this paper. All the major and commonly used commands would be outlined below for reference:

a) Creating a pool

We can create a pool named mypool with two disks c1d0 and c2d0 as mirrors

```
# zpool create mypool mirror c1d0 c2d0
```

Pools can also be created from normal regular files. This feature is especially useful when we are trying out zfs and we don't have free disks to spare.

```
# zpool create mypool mirror /space/mydisk1 /space/mydisk2
```

b) Creating filesystems

Once the pool is created we can create filesystems on top of the pool. Filesystems created on a pool shared the whole storage unless otherwise specified with quota and reservation limits.

We can create a home directory filesystem

```
#zfs create mypool/home
#zfs set mountpoint=/export/home mypool/home
```

Now we can created home directories for different users

```
#zfs create mypool/home/vineeth
```

```
#zfs create mypool/home/jan
#zfs create mypool/home/martin
#zfs create mypool/home/Jiri
```

c) Expanding the pool

When we get more storage, we can simply expand our pool by adding more disks.

```
#zpool add mypool mirror c3d0 c4d0
```

d) Setting the Properties

ZFS allows you to configure your filesystem by setting different properties exported by the filesystem. You can set the mount points, nfs shares, quotas, reservations, compression details, encryption details and more... to list a few examples:

To NFS-export all home directories:

```
#zfs set sharenfs=rw mypool/home
```

To Turn on the compression:

```
#zfs set compression=on mypool
```

To set quotas

```
#zfs set quota=10g mypool/home/vineeth
```

To set reservations:

```
#zfs set reservation=20g mypool/home/jan
```

e) Creating ZFS snapshots

Snapshots are read-only copy of a filesystem at a point in time. You can create a snapshot of the filesystem, before doing something nasty. In case of any emergencies, you can always revert back to the snapshot. This snapshot doesn't consume any additional space and it starts to consume space only when the actual dataset is modified. ZFS snapshots are accessible through the .zfs/snapshot directory in root of each filesystem.

Take a snapshot of jiri's directory:

```
#zfs snapshot mypool/home/jiri@May-1-2008
```

To roll back to a previous snapshot:

```
#zfs snapshot rollback mypool/home/jiri@April-1-2008
```

Access a file in a particular snapshot:

```
#cat ~jiri/.zfs/snapshot/April-1-2008/sample.c
```

f) Creating ZFS clones

Clones are writable copies of snapshots. This is also an instantaneous process and you can created unlimited number of clones.

To create a clone of opensolaris source code:
```
#zfs clone mypool/solaris@May-1-2008 mypool/ws/source-osol
```

g) Data backup and restore

You can backup and restore data using zfs send/receive commands. This is also powered by snapshots and is really fast and efficient.

Generate a full backup:
```
#zfs send mypool/solaris@May-1-2008 >/backup/osol-backup
```

Generate an incremental backup:
```
#zfs send -i mypool/solaris@day1 mypool/solaris@day10
>/backup/sosl-backup-1-10
```

Remote replication:
```
#sfs send -i mypool/solaris@day1 mypool/solaris@day10 |
ssh host zfs receive -d /mypool/solaris
```

h) Data Migration

You can export filesystem from one host to other transparently. As ZFS is endian independent, this works across different architecture also

Export pool from old server:
```
old-srv# zpool export mypool
```

Physically move the disks to new server and import the pool:
```
new-srv# zppol import mypool
```

i) ZFS and Zones

ZFS when combined with the zones(virtualization concept in solaris) can offer a lot more features. You can have a dataset delegated to zones which becomes secure as physical devices cannot be accessed inside zones. If the zone filesystem itself is in zfs, then zone creation becomes instantaneous by using snapshots and clones feature.

j) ZFS Root

OpenSolaris has zfs root by default. This has enhanced the usability of Solaris considerably:

- patching becomes more safer. You can revert back to a snapshot if patch was a bad patch
- Root filesystems has all the goodies that zfs has to offer — compression, encryption, checksumming, snapshots etc.
- Live upgrade is much more faster
- Grub is modified to understand ZFS

6 Conclusion

ZFS ends the suffering of sys-admins where they have a simple to use, feature rich powerful filesystem at their disposal now. ZFS is getting recognized for its efficiency, features and simplicity that it has been ported to other Operating Systems like Mac OS X and BSD. We can see that the caption 'Last word in filesystem' is no exaggeration when we start to understand this Filesystem and appreciate its features. As mentioned before, this paper is by no means a complete reference to ZFS and it is out of scope to explain ZFS in a short paper. Readers are encouraged to go online for more details and ofcourse have a hands-on to see the ZFS in action!

References

[1] http://www.sun.com/2004-0914/feature/

[2] ZFS Administration guide:
http://www.opensolaris.org/os/community/zfs/docs/zfsadmin.pdf

[3] http://www.opensolaris.org/os/community/zfs/docs/zfs_last.pdf

[4] http://http://www.sun.com/bigadmin/features/articles/zfs_part1.scalable.jsp

SAM-QFS

Jan Kopřiva

E-mail: jan.kopriva@sun.com

Abstract

Due to the new government laws, company policies and increased storage demands, data and its archives became a hot topic in the IT industry. Interesting problems emerged in the storage world: high performance data access, backup and recovery windows problem and high availability to name a few. Many storage products that appeared on the market tries to address those issues with variable rate of success. The goal of this paper is to introduce one such product, SAM-QFS of Sun Microsystems, which brings interesting solution to some of the mentioned problems.

1 Introduction

SAM-QFS is a recently open-sourced product of Sun Microsystems. As the name suggests, it consists of two parts: SAM and QFS.

QFS is a full-featured UNIX file system which can be configured also as a distributed, high performance file system. It is designed with scalability in mind and does not put any restrictions on the amount of information that can be managed. File system can easily handle data volumes that grows over time from Terabytes to Petabytes without any performance issues. SAM or *Storage Archival Manager*, enhances QFS of *Hierarchical Storage Management* (HSM) capabilities. In the following sections, we will discuss each part separately and introduce related features.

2 QFS

QFS is a 64-bit SAN (Storage Area Network) based filesystem. It does not mean one can not configure QFS on a local disk, but SAN disks are popular and common underlying storage for QFS. Because more than one device is usually connected to SAN, QFS offers volume management capabilities to gather disks or disk partitions into logical volumes. On the other hand, we can not configure software mirror with QFS. Other volume management software like SVM, Veritas or zvols have to be used for that.

2.1 File Allocations and Striped Groups

The volume management enables QFS to use two types of file allocations: *round-robin* and *striped* allocation. Round-robin allocation method writes one data file at a time to each successive device in the file system. When the file exceeds the size of the device, only the portion of the file is written to that device and the reminder goes to the next one.

Striped file allocation spreads the file data evenly across all of the file system devices or across the devices that belongs to the dedicated *striped group*. Striped group is a collection of devices within the filesystem and there can be up to 128 striped groups configured in QFS. Striped groups allow one file to be written to and read from two or more logical disk devices. The reason to have striped groups is performance. They enable very large disk allocations in parallel for fast sequential I/O. An example of storage group configuration could be a media system setup where video and audio files are stored. Video files are larger and have higher performance demands. They are perfect candidates to have a dedicated large striped group configured for them. Audio files are not that demanding and can reside on smaller striped group.

2.2 Paged vs. Direct I/O

QFS supports two I/O methods: *Paged I/O* and *Direct I/O*. Paged I/O is a default I/O method in QFS and when used, user data are cached in virtual memory before going to be written to disk. During Direct I/O, on the other hand, the user data goes directly to the underlying storage. For large sequential aligned I/O the Direct I/O performs better than Paged I/O.

2.3 Metadata

File systems use metadata to reference files and directories. In typical file systems the metadata resides on the same device as file data. The same holds true for QFS, but QFS is usually configured with metadata on separate dedicated partitions or disks. The advantage is that disk head movement is reduced during I/O operations and rotational latency is decreased as well.

File information is stored in inodes. Each inode occupies 512 bytes of disks space and is catalogued in .inodes file under the mount point. Usually, in the traditional UNIX file system, the number of pre-configured inodes limits the number of files in the file system. There is no such restriction in QFS. The file system allocates inode space dynamically. Therefore, to increase the number of files we just need to add more metadata devices. .inodes file can hold up to 4 billion of inodes, however it is recommended to have no more than 10 million of files in single QFS filesystem.

2.4 Disk Allocation Unit

The disk space is allocated in units called DAU or *Disk Allocation Unit*. In other words, this is the minimum amount of contiguous space QFS works with. The DAU is adjustable and is set during file system creation. It can not be changed dynamically later on. The benefit of having adjustable DAU is to tune the file system to the underlying storage. Especially, to minimise the overhead of read-modify-write operations typical for RAID volumes.

2.5 Shared QFS

As we already said, QFS can be configured as a full-fledged distributed file system that scales up to 256 nodes. In shared QFS environment, data flows through the attached SAN fabric, but messages and control information are exchanged through network. This is in contrast to other distributed file systems like Coda, where everything flows over the network.

In the shared QFS environment, one host acts as a metadata server and others are configured as clients. There can be more than one metadata servers configured, but at a given time, only one of the hosts acts as the metadata server. The others are potential metadata servers sometimes referred to as *metadata clients* (MDC). Those step in when the primary metadata server fails.

Shared QFS file system software is not only for Solaris. Linux is supported within the shared file system environment too. However, Linux hosts can act as clients only.

When a host makes changes inside the shared QFS environment, the file system assures that those changes are visible to other hosts immediately. To achieve this QFS, implements so called *lease* protocol. Basically, each hosts which intends to access a file, including the metadata server, must acquire an appropriate lease before the access. A granted lease gives the host privilege to perform certain operations on the file like read, write, append, truncate, lock, stage and open.

Hosts request a lease by sending a message to the server, which responds by either granting the lease or deferring the request. Lease requests are deferred if conflicting leases exist. For instance, only one write lease can be active unless multi-host write has been enabled.

The multi-host write is an ability for two or more nodes to write to the same file at the same time. File system relaxes the lease policy at this time and it is up to the application to ensure the data integrity. Also, to prevent data corruption applications need to use buffers aligned on page boundaries with multi-host writes.

In the usual case, a client expires its lease at given expiration time. When this happens, the client notifies the server and server may then notify any waiting clients that they may re-apply for their leases.

If a client fails, the server forcibly expires client's lease. This happens after additional time has been granted to the client, to allow for a heavily loaded client, network retransmission problems, etc.

3 SAM

Storage Archive Manager or SAM, enhances QFS with hierarchical storage management (HSM) ability. In general, HSM is a storage technique that migrates the data from high-cost media to low-cost media. HSM is sometimes referred to as multi-tiered storage. The critical or high priority data is stored on the first tier which is usually formed by fast and expensive fibre channel (FC) disk arrays. The first tier is sometimes referred to as *disk cache*. The less critical data are put onto other tiers to cheaper SATA disks, tapes or optical media. The number of tiers can depend on the amount of data being managed and the policy set in the storage environment. Typical system has three tiers that consist of FC disk arrays as the first tier, SATA disks as second and tapes as third tier.

3.1 Daemons and Concepts

SAM manages data in a slightly different way than other HSM products. SAM is not stand-alone product, instead it becomes an integral part of the QFS file system after its configuration. SAM brings additional storage tiers to the file system, which means that the file system namespace can span across additional disks, tape and optical libraries. SAM consists of multiple user-space daemons that receives events from the file system and communicate among themselves.

SAM does not migrate data, it creates *copies* instead. There can be up to four copies of data in the file system. A daemon called *sam-archiverd* is responsible for making the archive copies. When a file is created or changed in the file system, archiver is notified about that and looks up the archive policy specified for that file system. According to that policy, if the file is a candidate for archiving, its copy is made immediately. The archive copies represent file data backups, but archiver can also do metadata backups. Everything is done on-the-fly and file system does not need to be unmounted for doing that. This approach eliminates the well known backup window.

File data are put into the TAR format and stored on archive media. There are two types of archive copies recognised *disk archive* and *tape archive*. Disk archives are put to the existing filesystems, now raw disk is used for archivation. Every file system that can be mounted under Solaris can act as place holder for disk archives.

Archiver usually creates archives on SCSI or FC tape libraries attached to the host where the daemon runs. In case of shared QFS configuration archiver

and other SAM daemons have to run on metadata server. Clients can not archive directly. However, archiving can also be performed via network to remote disks and tapes. SAM implements a communication protocol to talk to ACSLS — the software that controls the remote tape library.

Archive copies can be *released* from its media manually or automatically according to the policy set by system administrator. For example data is released after some time or when the storage is filled over the predefined limit and more space is needed for important data. SAM storage policies can be quite sophisticated and the configuration is out of scope of this paper.

When a file is archived, optionally released and later on its data is needed on the disk cache, *sam-stagerd* daemon *stages* the file back. Note, that we can stage from any copy of the given file. The staging process is fully transparent to the applications. Staging the file back is actually the same operation as file data recovery. Again, it is performed on-the-fly therefore the recovery windows problem is minimised.

Media with old data that are no longer needed are *recycled* by *sam-recycler* command. This command follows the recycling policy set by the system administrator.

4 Conclusion

We have briefly introduced the most interesting SAM-QFS features. We did not mention how SAM-QFS can be made highly available, since this topic is quite extensive and the additional product that is used to achieve this — Sun Cluster — is not yet fully open-sourced.

SAM-QFS is still being developed. The next release 5.0 shall be out next year and brings, among others, a couple of redesigned SAM components, file system journaling, and file system online grow and shrink capability. Also, QFS is being enhanced for future HPC market. It will support intelligent storage nodes and object storage disks.

Reference

[1] docs.sun.com, *Sun StorageTek QFS File System Configuration and Administration Guide*, April 2007.

Implementace souborového systému v operačním systému HelenOS

Jakub Jermář

E-mail: jakub@jermar.eu

Úvod

Operační systém HelenOS je softwarový projekt, který z iniciativy studentů již několik let vzniká na půdě Matematicko-fyzikální fakulty Univerzity Karlovy v Praze. Cílem projektu není nahradit mainstreamové operační systémy, ale vytvořit pokud možno co nejlepší nosič pro zkoušení různých nápadů v oblasti implementace a výzkumu operačních systémů. Důkazem toho je například pět diplomových prací a jedna disertační práce, které jsou v současné době na fakultě řešeny a které s projektem HelenOS tematicky úzce souvisí. V minulosti byl HelenOS několikrát využit v kurzu operačních systémů jako projekt pro semestrální práce studentů. Smyslem všech těchto aktivit je postupně doplnit chybějící důležité vlastnosti pomocí více či méně obvyklých postupů. Charakteristickým rysem dosavadního vývoje je také úsilí o co možná nejobecnější vnitřní rozhraní a celkovou přenositelnost, což se projevuje tím, že HelenOS je možno spustit na celkem sedmi procesorových architekturách, počínaje procesory ARM a konče 64-bitovými procesory UltraSPARC.

Z různých reakcí, které se nám od zveřejnění projektu v roce 2006 dostaly, bylo patrné, že HelenOS je potenciálně velmi zajímavý projekt pro různé skupiny lidí – jako embedded operační systém, jako systém pro různá fyzikální měření, jako výukový systém nebo prostě jako zajímavá hračka. Uvědomili jsme si ale, že pro vznikající komunitu je složité se do jeho vývoje zapojit díky příliš velké adopční bariéře. Tato adopční bariéra byla z části tvořena skutečností, že pro vývoj HelenOSu je zapotřebí instalace simulátorů různých architektur a pro spolehlivý překlad ze zdrojových kódů také speciálních cross-kompilerů a cross-linkerů. Největší bariérou byla bez pochyby absence souborového systému, díky níž bylo obtížné uvědomit si potenciál, který HelenOS skrývá, a která značně omezovala použitelnost systému. V polovině roku 2007 tedy začaly snahy situaci napravit a navrhnout jednoduchý a zároveň rozšiřitelný rámec pro podporu souborových systémů v operačním systému HelenOS. Ve zbytku tohoto článku se budeme věnovat popisu toho, čeho bylo zatím dosaženo, s jakými implementačními problémy jsme se setkali a co na nás teprve čeká.

Pohled pod pokličku

V této části trochu nakoukneme HelenOSu pod pokličku. Zvláštní pozornost budeme věnovat návrhu IPC, protože ten má zásadní vliv na návrh samotného subsystému systému souborů.

Struktura systému

HelenOS obsahuje poměrně pokročilé mikrojádro, které vytváří příznivé prostředí pro běh uživatelských úloh (*task*) a umožňuje těmto úlohám navzájem komunikovat. Protože jsou adresové prostory jednotlivých úloh navzájem izolovány, jediným způsobem komunikace je buď prostřednictvím zasílání IPC zpráv a nebo explicitním sdílením paměti. Některé uživatelské úlohy jsou v jistém smyslu speciální, protože zastávají úkoly, které by mělo v tradičním operačním systému na starosti jádro. Jedná se především o ovladače zařízení a o systémové služby jako je například virtuální konzole nebo právě souborový systém. Jádro těmto speciálním úlohám propůjčuje vyšší privilegia a umožňuje jim tak například přímý přístup k určitému hardware.

Vlákna a vlákénka

Uživatelské úlohy mohou být v HelenOSu vícevláknové, což umožňuje jádru využívat případného paralelizmu víceprocesorového systému tím, že naplánuje dvě vlákna (*thread*) jedné úlohy na dva různé procesory. HelenOS tedy podporuje a plánuje jaderná vlákna. Ta mohou mít, a v drtivé většině případů mají, i uživatelský kontext. Z důvodů popsaných níže v části o IPC se v uživatelském prostoru za podpory standardní knihovny každé jaderné vlákno rozpadá ještě na jedno až několik vlákének (*fibril*). Tato vlákénka běží v kontextu svého vlákna a jádro o nich nic neví a nedokáže je přímo plánovat. To má na starosti právě již zmíněná standardní knihovna. Na rozdíl od jaderných vláken, která mohou být v důsledku preempce pozastavena a odebrána z procesoru prakticky v libovolný okamžik, vlákénko běží, dokud se samo nevzdá řízení nebo dokud nezavolá IPC operaci, která nemůže být okamžitě provedena.

IPC a asynchronní framework

HelenOS navzdory moderním trendům využívá převážně asynchronní komunikaci, avšak podporuje i synchronní. Názvosloví užité v celém IPC subsystému je odvozeno z přirozené abstrakce telefonního spojení mezi člověkem na jednom konci a záznamníkem na konci druhém. Asynchronnost je zde dána právě tím, že na straně volaného je záznamník a nikoliv další člověk, který by byl s to okamžitě odpovídat.

Běžná IPC komunikace probíhá následujícím způsobem. Uživatelské vlákénko použije jednu ze svých telefonních linek (*phone*), kterou má propojenu se záznamníkem (*answerbox*) cílové úlohy, a uskuteční po ní krátký hovor (*call*). Odesílající vlákénko může buď pokračovat ve své práci a nebo počkat na odpověď. Cílová úloha má nyní ve svém záznamníku zmeškaný hovor. Dříve či později dojde k jeho vyzvednutí nějakým vlákénkem patřícím do cílové úlohy, které zprávu zpracuje, odpoví na ni a nebo ji přepošle nějaké třetí straně. V každém případě nakonec někdo na zprávu odpoví. Odpověď je analogicky uložena do záznamníku, v tomto případě záznamníku úlohy, která uskutečnila původní hovor.

Životní cyklus jednoho asynchronního IPC požadavku v sobě skrývá nejedno úskalí. V případě, že úloha obsahuje více než jedno vlákénko, není úplně zřejmé, pro které z nich je určen hovor uložený na záznamníku úlohy. Problém je tak závažný, že nám nabourává koncept spojení, jak ho například známe z BSD socketů. Krom toho obsluha asynchronní komunikace většinou vede k nepřehledné změti různých callbackových funkcí a kódu, který si pomocí stavových proměnných snaží udržet přehled o tom, kdo zrovna čeká na jakou zprávu či odpověď.

HelenOS tyto problémy řeší nasazením specifického asynchronního frameworku a intenzivním využitím manažerských vlákének. Speciální manažerská vlákénka jsou jako jediná oprávněna vyzvedávat hovory přímo ze záznamníku a dále je předávat uživatelem definovaným běžným vlákénkům, která obsluhují dílčí spojení. Pokud se běžné vlákénko rozhodne čekat na odpověď, která ještě není k dispozici, asynchronní framework sám zaregistruje potřebný callback pro zachycení odpovědi a přepne se na jiné vlákénko, které může pokračovat v běhu. Na původní vlákénko se opět přepne až tehdy, když už je odpověď na jeho IPC požadavek připravena. Z pohledu jednotlivých vlákének vypadá komunikace prostřednictvím asynchronního frameworku velmi přímočaře a jednoduše.

IPC a sdílení paměti

Jak již bylo řečeno, uživatelské úlohy a v nich obsažená vlákénka mohou komunikovat pomocí ukládání krátkých hovorů na záznamník volané úlohy. To, že zde hovory označujeme jako krátké má svůj doslovný význam, protože tato metoda komunikace dokáže přenést pouze velmi malé množství dat. Původně se jednalo o čtyřnásobek toho, co daná architektura uměla uložit do všeobecného registru. Pro zasílání delších a velmi dlouhých zpráv nebyl takovýto způsob komunikace přijatelný. Již od začátku jsme plánovali, že naše jádro bude podporovat sdílení paměti mezi jednotlivými úlohami. Na konec jsme sdílení paměti dovedli takřka k dokonalosti tím, že jsme jej plně zaintegrovali do IPC. Ukázalo se totiž, že pomocí krátkých IPC zpráv je možno velmi elegantně dohodnout podmínky sdílení mezi jednotlivými účastníky. Pro sdílení paměti použije úloha speciální IPC

metodu, *IPC_M_SHARE_IN* nebo *IPC_M_SHARE_OUT*, resp., a v argumentech zprávy uvede údaje o zdrojové nebo cílové, resp., virtuální adrese, velikosti sdílení a případně ještě informace o typu sdílení. Jádro si všimne, že jedna úloha iniciuje sdílení paměti s jinou úlohou a začne sledovat vývoj událostí. Druhá úloha obdrží zprávu a na základě obdržených informací se rozhodne, zda na sdílení přistoupí či nikoliv. Pokud přistoupí, odpoví první úloze kladným návratovým kódem. Jádro kladný návratový kód zaznamená a provede kroky vedoucí k vytvoření již dohodnutého sdílení. Pokud k dohodě nedojde, jádro zůstane v této záležitosti pasívní a iniciátor obdrží chybovou návratovou hodnotu.

Návrh a implementace

Funkcionalita celého subsystému systému souborů je rozložena do tří samostatných celků:

- podpora ve standardní knihovně,
- podpora ve VFS serveru,
- implementace výstupního VFS protokolu v ovladači koncového souborovém systému a
- podpora v knihovně libfs.

Standardní knihovna

Standardní knihovna obsahuje kód, který převádí více méně POSIX volání dané uživatelské úlohy na protokol, kterému rozumí vstupní část VFS serveru a tímto způsobem mu předává klientské požadavky. Některá volání, jako například *opendir()*, *readdir()*, *rewinddir()* a *closedir()* implementuje standardní knihovna přímo pomocí jiných volání – v tomto případě tedy *open()*, *read()*, *lseek()* a *close()*. Protože VFS server umí pracovat pouze s absolutními souborovými cestami, zajišťuje standardní knihovna rovněž volání *getcwd()* a *chdir()* a také převod všech relativních cest na absolutní. Předávání absolutních cest nemusí být sice vždy nejefektivnější, zato přináší řadu výhod. Jednak velmi zjednodušuje návrh VFS serveru a knihovny libfs a také umožňuje lexikální zpracování všech výskytů komponenty *tečka-tečka*. Kromě výše uvedeného, standardní knihovna neobsahuje žádné datové struktury a algoritmy pro podporu systému souborů. Každý požadavek, který nedokáže vyřešit sama obratem přeposílá formou jedné nebo několika IPC zpráv VFS serveru.

VFS server

VFS server je ústředním a nejsložitějším prvkem podpory souborových systémů v operačním systému HelenOS. Představme si pomyslné rozdělení tohoto serveru na část vstupní a výstupní.

Část vstupní přebírá požadavky od klientských uživatelských úloh. Parametry těchto požadavků jsou buď absolutní cesty k souborům nebo deskriptory již otevřených souborů. Pokud je parametrem cesta, provede VFS operaci *vfs_lookup_internal()*, jejímž výsledkem je tak zvaný VFS triplet. VFS triplet je uspořádaná trojice, která pomocí globálního čísla souborového systému, globálního čísla zařízení a čísla souboru na dané instanci souborového systému jednoznačně určuje nějaký soubor. Zahešováním podle VFS tripletu se VFS server pokusí najít odpovídající VFS uzel. VFS uzel, instance typu *vfs_node_t*, je abstrakce, která v adresovém prostoru VFS serveru představuje nějaký soubor, na který má VFS referenci. Všechny soubory, se kterými VFS pracuje jsou reprezentovány pomocí VFS uzlů. V případě, že parametrem operace je deskriptor souboru, je převod na VFS uzel daleko jednodušší, protože tabulka otevřených souborů, kterou VFS udržuje pro každou klientskou úlohu zvlášť, přímo obsahuje ukazatel na strukturu reprezentující odpovídající otevřený soubor (*vfs_file_t*) a ta zase přímo ukazuje na VFS uzel.

Jakmile VFS server zná VFS uzel, kterého se VFS operace týká, může ji předat ovladači příslušné instance souborového systému a nebo ji provést sám. V současné době realizuje VFS server například operaci *VFS_SEEK*, protože *VFS_SEEK* nevyžaduje další interakci s příslušným souborovým systémem. *VFS_SEEK*, která odpovídá POSIX volání *lseek()*, pouze posune ukazatel aktuální pozice pro čtení a zápis v příslušné struktuře *vfs_file_t*. K tomu může za určitých okolností potřebovat znát velikost daného souboru – tu ale VFS server udržuje přímo v odpovídajícím VFS uzlu.

Výstupní část VFS pak komunikuje s ovladačem koncového souborového systému, jehož číslo a také číslo zařízení, na kterém je tento souborový systém připojen, zná díky informacím uloženým ve VFS uzlu. Soubor všech zpráv, které může VFS server poslat jednotlivým ovladačům koncových souborových systémů jednoznačně definuje výstupní protokol VFS. Všechny serverové úlohy, jejichž ambicí je ovládat nějaký konkrétní souborový systém, musí tomuto protokolu rozumět a musí jej implementovat. V podstatě se jedná o objektově orientovaný návrh, který má jen trochu přirozenější formu než VFS v monolitických kernelech, které bývají realizovány pomocí sady ukazatelů na funkce.

PLB a kanonické cesty

Významnou úlohou, na jejímž řešení se podílí VFS server spolu s připojenými souborovými systémy, je převod znakových řetězců reprezentující cesty k sou-

borům na VFS triplety. VFS server postupuje zhruba tak, že se postupně ptá ovladačů souborových systémů připojených podél dotyčné cesty. Každý dotázaný ovladač vyřeší maximální část cesty od již vyřešené části až po případný další bod připojení (*mount point*) jiného souborového systému. Cestu nakonec vyřeší poslední dotázaný souborový systém a pošle VFS serveru požadovaný VFS triplet jako odpověď. Cílem návrhu podpory souborových systémů v operačním systému HelenOS bylo vyhnout se situaci, ve které by se cesta či její část neustále kopírovala mezi VFS serverem a dílčími ovladači koncových souborových systémů. Za tímto účelem alokuje a udržuje VFS server cyklickou vyrovnávací paměť, do které umisťuje vyhledávané cesty. Každá implementace koncového souborového systému tuto vyrovnávací paměť, zkráceně označovanou jako PLB (*Pathname Lookup Buffer*), s VFS serverem sdílí v režimu pro čtení. Samotné umístění cesty do PLB a řízení distribuovaného vyhledání je implementováno v již zmíněné funkci *vfs_lookup_internal()*.

vfs_lookup_internal() musí garantovat, že do PLB uloží jen cesty v kanonickém tvaru. To znamená, že musí ověřit že cesta neobsahuje například dvě lomítka za sebou, že v cestě není žádný symbol *tečka* či *tečka–tečka*. Pokud není cesta v tomto smyslu v pořádku, použije se speciální restartovací automat, který ji postupným opracováváním převede na kanonický tvar.

VFS server tedy uloží již kanonickou cestu do PLB a zkontaktuje souborový systém, který je připojen ke kořeni adresářového stromu. Zašle mu zprávu typu *VFS_LOOKUP* a jako argumenty uvede indexy prvního a posledního znaku cesty v PLB. Poté, co kořenový souborový systém vyřeší svou část cesty, nedopoví nutně VFS serveru. Pokud je totiž ještě co řešit, přepošle zprávu *VFS_LOOKUP* tomu souborovému systému, na jehož bodě připojení sám skončil vyhledávání a upraví argumenty zprávy tak, aby index prvního znaku cesty v PLB ukazoval na první nevyřešený znak. Vyhledávání v tomto duchu pokračuje až se nakonec jednomu souborovému systému podaří cestu vyřešit celou. Teprve tento poslední souborový systém odpoví na původní zprávu *VFS_LOOKUP* a v odpovědi zahrne celý VFS triplet odpovídající nalezenému souboru. Díky právě popsanému návrhu ušetří HelenOS zhruba polovinu systémových volání, nemluvě o ušetřeném kopírování paměti.

Ovladač koncového souborového systému

Jak již bylo řečeno, každý ovladač souborového systému musí implementovat výstupní VFS protokol. Pro účely ověření funkčnosti celého konceptu jsme přistoupili k návrhu a realizaci velmi jednoduchého souborového systému, jehož datové struktury existují jen v paměti počítače a který nemá ani žádný diskový formát – TMPFS. Na druhou stranu, TMPFS má hierarchickou adresářovou strukturu a umožňuje plnohodnotně otestovat všechny souborové operace, které mohou uživatelské úlohy používat.

Pro potřeby složitějších souborových systémů, které mají diskový formát, jsme navrhli speciální server – DEVMAP, který dokáže propojit příslušný ovladač koncového souborového systému s ovladačem blokového zařízení podle globálního čísla zařízení. Potřeba komunikace mezi souborovým systémem a blokovým zařízením je jasná. Soubor zpráv, které bude ovladač souborového systému ovladači blokového zařízení zasílat jednoznačně určí protokol pro přístup k blokovým zařízením a, podobně jako v případě výstupního VFS protokolu, umožní používat libovolná bloková zařízení s ovladači podporujícími tento protokol.

Kromě implementace výstupních VFS operací musí ovladače koncových souborových systémů implementovat také rozhraní knihovny *libfs*, kterou popisujeme níže.

Knihovna libfs

Některé konstrukce, které logicky patří do ovladače koncového souborového systému, by se objevovaly v implementaci každého takového ovladače – někdy jen s minimální obměnou, někdy dokonce na vlas stejné jako v ostatních implementacích. Pro odstranění tohoto problému byla založena knihovna *libfs*, jejímž cílem je pomocí vhodných abstrakcí koncentrovat podobné duplicity do jednoho místa. V současné době je součástí knihovny kód, kterým se jednotlivé souborové systémy registrují u VFS serveru a dávají mu vědět o své existenci.

Dalším a velmi podstatným příkladem *libfs* kódu je funkce *libfs_lookup()*. Tato funkce je vlastně generickou šablonou pro implementaci výstupní VFS operace *VFS_LOOKUP*. Tuto operaci musí podporovat každý koncový souborový systém, ale vzhledem k její složitosti a také provázanosti mezi místy připojení jednotlivých souborových systémů je strategicky výhodnější spravovat její kód centrálně. *libfs_lookup()* je specifická také tím, že neimplementuje pouze vyhledávání, ale rovněž vytváření a rušení jmen souborů v adresářovém stromě, vytváření prázdných souborů a adresářů a určování rodičovského VFS uzlu k uzlu zadaného cestou. Aby to fungovalo v rámci konkrétního souborového systému, musí tento implementovat několik operací, které *libfs* řeknou, jak přesně má prohledávat adresář, jak vytvořit či zrušit jméno souboru v adresářovém stromě nebo jak vytvořit či smazat soubor.

Změny vynucené implementací

Předchozí část se věnovala stručnému popisu toho, čeho bylo zatím na úrovni podpory souborových systému v operačním systému HelenOS dosaženo. V této části se podíváme na několik změn, které si tato podpora vynutila.

Vylepšení IPC

Některé operace VFS serveru by bylo hezké provést zasláním jedné krátké IPC zprávy, pokud by se ovšem vystačilo s dostupnými IPC argumenty. Příkladem takové operace je již zmiňovaná *vfs_lookup_internal()*. Tato interní funkce zasílá kořenovému souborovému systému zprávu *VFS_LOOKUP* o pěti argumentech, což je o dva více, než bylo možné v dřívější implementaci předcházející přidání podpory souborového systému. Tento případ jasně ukazuje, že na rozdíl od jednodušších protokolů, které HelenOS podporoval již dříve, protokol složitostí odpovídající potřebám protokolu VFS vyžaduje větší počet argumentů IPC zpráv. Omezení počtu IPC argumentů má své kořeny v počtu argumentů, které lze předat v registrech procesoru během systémového volání. Zvýšením tohoto limitu na šest jsme docílili toho, že při asynchronním typu IPC je nyní možno předat čtyři IPC argumenty přímo v registrech. Neoptimalizovaná verze IPC dokáže předat až pět IPC argumentů, s výhledem na další rozšíření do budoucna.

Pro komunikaci mezi standardní knihovnou a VFS serverem, během níž se předávají řetězcové argumenty, by bylo použití sdílené paměti příliš těžkopádné. Ze své povahy převážně jednorázové přesuny dat je mnohem lepší uskutečnit nikoliv sdílením, ale kopírováním. Kopírování vyznívá lépe i pro operace *read()* a *write()*, protože se opět jedná o jednorázový přesun dat mezi klientskou úlohou a ovladačem koncového souborového systému. Tato úvaha vedla k zavedení dvou nových systémových IPC metod, *IPC_M_DATA_READ* a *IPC_M_DATA_WRITE*, jejichž použití je analogické použití metod pro sdílení paměti. Rozdíl je samozřejmě v tom, že jádro při použití těchto metod nenastaví sdílení paměti, ale fyzicky zkopíruje blok dat z jednoho adresového prostoru do druhého.

Příklad *read()* a *write()* naznačil, že IPC zprávy mohou přejít přes jeden či několik mezilehlých úloh – v tomto případě VFS server. Pro podporu přeposílání kopírovacích zpráv a zpráv pro sdílení paměti bylo potřeba toto chování povolit a zaručit, že úlohy, které takové zprávy přeposílají, nemohou do obsahu zprávy nijak zasáhnout. Příjemným vedlejším efektem tohoto malého vylepšení je skutečnost, že samotná kopie dat nebo sdílení paměti bude provedeno pouze jednou, a to mezi prvním odesílatelem a posledním adresátem příslušné zprávy.

Tématu přeposílání zpráv se týká i další změna IPC. Protože jsou jednotlivá IPC spojení díky asynchronnímu frameworku obsluhována samostatnými vlákénky, způsobilo by pouhé přeposlání nějaké IPC zprávy dalšímu adresátovi vytvoření nového vlákénka v úloze tohoto adresáta. Nové vlákénko by mělo na starosti spojení mezi původním odesílatelem a tímto posledním adresátem, což ovšem nemusí být to, co potřebujeme. Příklad, který toto chování ilustruje je implementace operace *VFS_READ*, odpovídající POSIX volání *read()*, uvnitř VFS serveru. VFS server obdrží od standardní knihovny jedné ze svých klientských úloh zprávu *VFS_READ*, díky čemuž pozná, že jako další bude následovat zpráva

typu *IPC_M_DATA_READ* obsahující pokyn pro nakopírování přečtených dat do klienta. Po jejím obdržení ji VFS server nemůže jenom tak přeposlat ovladači koncového souborového systému, protože by ji tento zařadil do jiného spojení. Místo toho ji VFS server při přeposlání označkuje, čímž dá dalšímu příjemci najevo, že zpráva prošla přes něj a že je ji tedy třeba chápat jako součást spojení mezi nimi. Označkování nemá žádný vliv na směrování odpovědi na zprávu – odpověď bude vždy doručena přímo původnímu odesílateli.

Závěr

Podpora souborových systémů obsažená v operačním systému HelenOS se vyznačuje přiměřenou jednoduchostí a přímočarostí. Zatím se jedná spíše o funkční prototyp než o plnohodnotný VFS, jaký lze nalézt v jiných, mnohem vyzrálejších operačních systémech, ale i přesto současná verze obsahuje několik zajímavých myšlenek. Bylo dosaženo částečné kompatibility s tou částí standardu POSIX, která se týká souborových systémů. Implementace zatím nepodporuje přístupová práva ani nezaznamenává čas posledního přístupu či modifikace souboru. Z našeho pohledu se však jedná o funkcionalitu sekundární, kterou bude možno přidat později. V některých případech se naše implementace mírně odklání od POSIXových chybových návratových kódů a některé okrajové situace, které v POSIXu znamenají chybu, zvládne ošetřit. Pokud jde o ovladače koncových souborových systémů, vytvořili jsme čistě paměťový souborový systém TMPFS, který ovšem dovoluje otestovat a vyzkoušet funkcionalitu celého VFS serveru. V současné době pracujeme na podpoře druhého koncového souborového systému – FAT16. Koexistence dvou souborových systémů nám umožní rozšířit VFS server a funkci *libfs_lookup()* tak, aby bylo možno připojit více souborových systémů současně. Pro diskové souborové systémy vyvstane potřeba vytvořit nějaké jednotné rozhraní pro vyrovnávací paměti diskových bloků. Postupným přidáváním dalších a dalších souborových systémů dojde k vyzrání a stabilizaci VFS protokolu.

Zda se nám tyto kroky podaří v budoucnu realizovat, záleží také do určité míry na schopnosti tvořící se HelenOS komunity přispívat do vývoje tohoto systému. Díky snahám nabídnout komunitě již použitelný základ systému by mělo být jednodušší stát se její součástí.