VYSOKÁ ŠKOLA EKONOMICKÁ V PRAZE

Fakulta informatiky a statistiky

obor podnikové informační systémy

Implementace protokolů TCP/IP v operačních systémech

Bakalářská práce

Jan Kotek

Vedoucí práce: Mgr. Lada Vronková

Prosinec 2006

Poděkování

Rád bych tímto poděkoval vedoucí své práce za podporu
při její tvorbě a shovívavost nad mou nepořádností.

Zadání:

Zpracujte přehled implementací TCP/IP protokolů v různých operačních systémech. Srovnejte tyto implementace z hlediska výkonu, bezpečnosti a množství podporovaných funkcí. Popište způsob detekce operačního systému na základě jejich síťového chování.

Prohlášení

Prohlašuji že jsem bakalářskou práci vypracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze kterých jsem čerpal.

V Galway dne 20. prosince 2006

Jan Kotek

Abstrakt

Podpora síťových protokolů je pro každý operační systém velice důležitá. Tato bakalářská práce porovnává implementace rodiny protokolů TCP/IP v různých operačních systémech. Práce se také zabývá historií a původem těchto implementací.

V práci je podrobněji popsána funkčnost implementace TCP/IP v Linuxu.

Pozornost je soustředěna na chyby v protokolech TCP/IP, které mohou být zneužity k útoku nebo získávání informací. V práci jsou popsány nejběžnější typy útoků a postupy jak se proti těmto útokům účinně bránit.

Práce se dále zabývá srovnáním implementací v různých operačních systémech. Jsou probrány různé možnosti testování. Výsledkem srovnání je přehled podporovaných vlastností a doporučení použití operačního systému.

Abstract

Support of network protocols is very important for any operating system. This bachalor thesis compare implementations of TCP/IP protocol family in various operating system. This bachalor thesis also concerns history and origin of this implementations

TCP/IP implementation in Linux is described in more details in this work.

The attention is further concentrated on mistakes in TCP/IP protocols, which can be used for attack or for information escape. The most common types of attacks and at the same time methods for an effective protection against the attacks are described.

Implementations in various operating systems are also compared in this work. Some types of test are described. The result of comparaition is list of supported features and recommendation of best operating system use.

Obsah

Úvod

Způsob jak různé operační systémy implementují protokoly z rodiny je svým způsobem fascinující. Mezi různými počítači jsou obrovské rozdíly a přesto se dokáží navzájem domluvit. Malinký jednočip, mobilní telefon, osobní počítač i superpočítač za několik miliard, používají stejný protokol pro vzájemnou komunikaci.

Tato práce zpracovává přehled různých implementací protokolů TCP/IP. Soustředil jsem se na nejpoužívanější serverové a operační systémy. Zkoumal jsem společné vlastnosti a odlišnosti těchto implementací. Jednotlivé implementace jsem analyzoval z hlediska bezpečnostního, výkonnostního a možného nasazení. Součástí práce je také přehled podporovaných vlastností jednotlivých implementací.

Přes velkou různorodost má většina současných implementací TCP/IP společného předka. Je jím 4.4BSD-Lite, tento operační systém obsahoval velmi kvalitní implementaci a přitom byl volně k použití. Většina operačních systémů tuto implementaci převzala. Tato práce mapuje historii těchto převzetí až do současných operačních systémů.

V práci podrobněji popisuji TCP/IP implementaci v operačním systému Linux. Je vysvětlen způsob jakým je implementace rozdělena do vrstev a jak spolu tyto vrstvy komunikují. Dále je popsána cesta paketu od linkové vrstvy přes síťovou a transportní vrstvu až k aplikační vrstvě.

V předposlední části se věnuji bezpečnostním problémům spojených s protokoly TCP/IP. Přímo v návrhu TCP/IP se vyskytují některé chyby, které lze zneužít k útoku i na plně zabezpečených systémech. Probírám nejčastější typy útoků a možnou obranu proti nim.

Poslední kapitola je věnována srovnání implementací v různých operačních systémech. Je popsáno několik kritérií testování a způsobů testování. Dále je popsáno praktické využití získaných poznatků pro výběr operačního systému. Kapitola dále obsahuje přehled podporovaných vlastností TCP/IP protokolů v různých operačních systémech.

V příloze se dále věnuji detekci operačního systému přes síť. Tato informace může být důležitá pro případný útok a proto je třeba vědět jak ji chránit. Popisuji různé způsoby detekce od jednoduchého dotazu až po sofistikované metody používané v programu NMAP.

Rád bych hned v úvodu vysvětlil některé termíny používané v této práci. První nejasnost by mohla vzniknout z hlediska síťových vrstev. Nepožívám zde názvosloví z OSI modelu (7 vrstev), ale vycházející z TCP/IP. Fyzická, linková, síťová a transportní vrstva jsou zachovány. Prezenční, relační (session layer) a aplikační jsou sloučeny do jediné aplikační vrstvy.

Další nejasnost by mohla vzniknout v termínu stack. Je to technické označení implementace protokolu. Například IP stack je implementace protokolu IP.

V závěrečné části mluvím o „všech operačních systémech“. Pod tímto označením myslím všechny operační systémy podrobněji popsané v této práci a široce

používané. Konkrétně tedy Linux 2.4, Linux 2.6, FreeBSD, OpenBSD, Windows NT 4, Windows 2000, Windows XP a Windows 2003.

Z přehledu jsem záměrně vypustil Windows 95, Windows 98 a Windows Millenium. Nejsou již výrobcem podporovány, jejich podíl rychle klesá a z technického hlediska jsou slepou uličkou. Další operační systém, který záměrně v přehledech nezmiňuji, je NetBSD. Vývoj tohoto systému je v krizi a z technického hlediska je velmi podobný OpenBSD. Do přehledu není také zahrnut Solaris a další komerční Unixy. Tyto systémy jsou velmi zajímavé, ale bohužel o nich nemám dostatek informací.

V práci jsou probrány také méně obvyklé implementace použitelné v elektronických zařízeních. Jejich počet bude s příchodem IPv6 stoupat a proto je považuji za důležité. Pro nedostatek informací jsou ale také vypuštěny ze závěrečného přehledu.

Protokoly TCP/IP

TCP/IP zkratka znamená (Transmission Control Protocol/Internet Protocol). TCP/IP jsou sice jen dva protokoly, ale obvykle se pod tímto označením myslí celá rodina protokolů.

Protokoly TCP/IP zabezpečuje propojení dvou programů. Propojení je transparentní na aplikační úrovni, programy se tedy nemusí starat o detaily přenosu. Díky tomuto odloučení je možné zavádět služby a protokoly na aplikační úrovni nezávislé na transportní vrstvě (například http a ftp protokoly mohou fungovat i nad UDP).

Protokoly jsou zakotveny v sérii RFC doporučení. Je to průmyslový standart, na jeho dodržování nedohlíží žádná centrální organizace, ale vzájemný konsenzus.

Vývoj protokolů TCP/IP

Rodina protokolů TCP/IP je potomkem studené války. Armáda USA měla strach z nukleárního útoku. Zničení několika komunikačních center by znemožnilo komunikaci v celé zemi a tím paralyzovalo stát a armádu. Proto americká armáda pověřila Paula Barana z organizace RAND vypracováním studie na decentralizovanou komunikační síť. Požadavek byl, aby každá buňka sítě byla nezávislá a síť byla schopná pracovat i po vyřazení většiny center. Baran navrhl několik řešení, ale jeho konečný návrh byla paketově spínaná síť.

V tomto typu sítě jsou data rozděleny na pakety. Každý paket je označen adresou odesílatele a příjemce. Na základě této informace je paket předáván z počítače na počítač, dokud nedojde k příjemci. Každý paket putuje po síti samostatně a pokud se cestou ztratí, je možné jeho vysílání zopakovat. Optimální cesta paketů se stanovuje za chodu, možných cest je více a pokud některá přestane fungovat,

prostě se zvolí jiná. Tato síť je díky množství cest odolná i proti nukleárnímu útoku.

První sítí tohoto typu byla výzkumná a testovací síť v Národní fyzikální laboratoři v Anglii. Grantová agentura Ministerstva obrany USA ARPA (Advanced Research Projects Agency) se rozhodla realizovat podobný projekt. Zárodek dnešního Internetu tak vznikl v roce 1968 pod názvem ARPANET.

ARPANET neměl zpočátku žádný společný protokol, každý spoj se realizoval jinak a připojení nového člena sítě bylo komplikované. V roce 1969 vznikl požadavek na univerzální protokol.. Zadáním bylo vyvinout protokol pro komunikaci v rozsáhlých počítačových sítích. Pro potřeby americké armády a univerzit vznikl protokol NCP (Network Control Protocol), tento protokol zpočátku zabezpečoval komunikaci v ARPANETu. Protokol NCP byl velmi jednoduchý a s růstem ARPANETu přestal dostačovat. Jeho přepracováním a rozšířením vznikla rodina protokolů TCP/IP.

TCP/IP vznikl jako výsledek dalšího projektu agentury DARPA, který měl za cíl zkoumat techniky a technologie pro propojování paketových sítí různých typů. První verze specifikací protokolu TCP/IP byla hotová v září roku 1973 a prezentovaná na konferenci na Univerzitě of Sussex. Publikovaná byla následujícího roku v časopise IEEE Transactions on Communications. Autory specifikace byli Bonton Cerf a Robert Kahn.

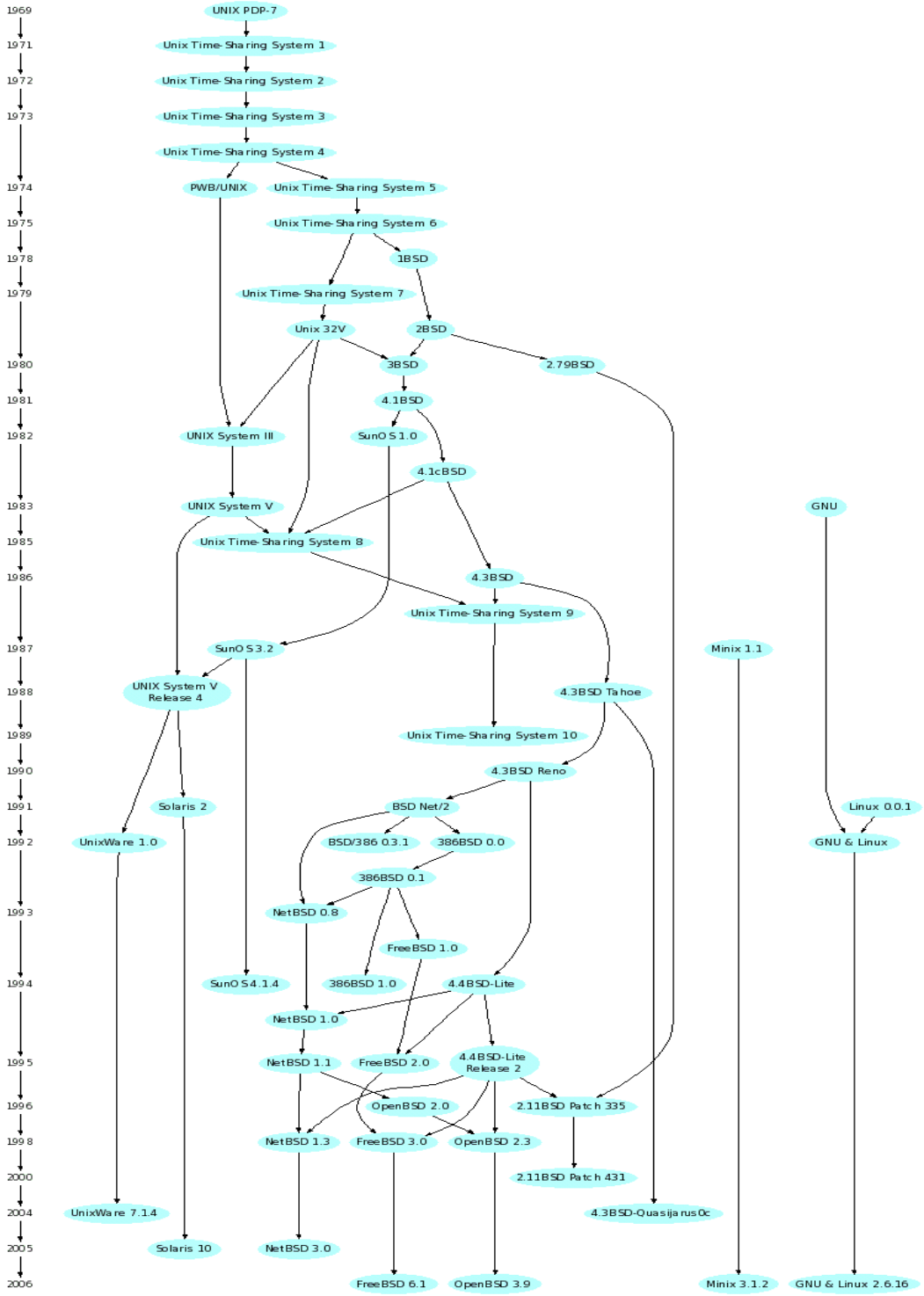
Práce na specifikaci probíhali na University College v Londýně, na Standford Univerzity a ve firmě BBN. V roce 1977 proběhly první praktické zkoušky. V letech 1978 až 1979 získává TCP/IP konečnou podobu.

TCP/IP používaly různé sítě k připojování do ARPANETu již od roku 1978. Oddělením vojenské části ARPANETu pod názvem MILNET v roce 1983, se

z ARPANETu stává civilní síť a rodina protokolů TCP/IP jejími výhradními komunikačními protokoly.

Na původní TCP/IP se postupně začínají nabalovat další, například Domain Name Service (DNS). Proces rozšiřování rodiny TCP/IP probíhá dodnes, díky nové generaci protokolů spojených s IPv6.

Původní TCP/IP implementace v BSD



Vývoj Unixových systémů. Autor Jean-Baptiste Campesato [2]

Většina současných implementací TCP/IP má jednoho společného předka. Je jím IP stack z Berkeley Software Distribution [1] [2], konkrétně z 4.4 BSD-lite. Z tohoto systému převzaly kód všechny současné *BSD. Z nich dále přebírají části implementace TCP/IP systémy Linux, Windows, MacOSX a další.

První distribuce Unixu z Bellových laboratořích obsahovaly zdrojový kód a umožňovali ostatním tento kód měnit a dále rozšiřovat. Toho využila univerzita v Berkeley a začala kód rozšiřovat a upravovat pro vlastní potřeby. První Unix, který se dostal do Berkeley byl PDP-14 vydaný v roce 1974. Jak v Berkeley přidávaly další a další funkce, ostatní univerzity vyjádřily o software v Berkeley zájem. Proto v roce 1977 student Bill Joy sestavil a rozeslal na páskách první Berkeley Software Distribution (1BSD) . Tato verze obsahovala pouze doplňkové programy k existujícím Unixům, neměla vlastní kernel. Také 2BSD ještě nebyla operační systém. Vývoj pokračoval verzí 3BSD v roce 1979 a 4BSD v roce 1980.

4.2BSD

V roce 1981 se z ARPANETu stávala rozsáhlá síť a připojovalo se na něj stále více univerzit. Protokol TCP/IP byl dobře specifikován v předběžné verzi, ale chyběla mu dobrá implementace. Na univerzitách byla BSD velmi rozšířena a také DARPA měla s Berkeley dobrou zkušenost z dřívější spolupráce. Volba pro implementaci protokolů TCP/IP padla na BSD. DARPA vypsala grant na rozšíření 4.1BSD. Součástí rozšíření měl být kromě implementace TCP/IP také rychlejší filesystém a celkové vyladění výkonu.

Vedením nového projektu byl pověřen Duane Adams. Programátoři se scházeli každý půlrok mezi dubnem 1981 a červnem 1983, na těchto schůzkách se probíral postup prací a přesné požadavky na funkčnost. Od roku 1984 tyto schůzky přerostly v konferenci na které se probíraly detaily TCP/IP.

První prototyp implementace TCP/IP byla napsána Robem Gurwitzem z firmy BBN. Tuto implementaci měl do BSD začlenit student Bill Joy. Během začleňování ale vyšly najevo problémy s touto implementací. Implementace od BBN totiž byla určena k práci na pomalých modemových linkách, zde si vedla velmi dobře. Pro rychlý Ethernet byla ale tato implementace nepoužitelná, neúnosně zatěžovala počítač a nedokázala proto využít plnou šířku pásma. Joy se tedy rozhodl implementaci přepsat a optimalizovat, tak aby dokázala pracovat i na rychlých linkách. Toto přepsání se poněkud zvrhlo. Začalo být jasné že TCP/IP nebude jediným protokolem a proto Joy oddělil implementaci protokolu od zbytku systému a ovladačů pomocí interfaců. Přidávání nových protokolů a hardware se tak podstatně zjednodušily. Toto rozseknutí také umožnilo používat více protokolů současně na jednom počítači, to je velmi podstatné pro propojení sítí různých typů.

Společně s implementací TCP/IP vznikaly programy které by nový síťový protokol dokázaly využít. Jednalo se o programy: rcp, rsh, rlogin a rwho. Tyto programy měli být pouze dočasné řešení.

První vývojová verze byla uvolněna pod názvem 4.1a BSD v dubnu 1982. Jako první Unix obsahovala jádro s implementací TCP/IP a síťové programy. Tato verze byla určena pouze pro vnitřní použití a měla být pouze testovací. Přesto se dostala ven a hojně se používala. Obsahovala množství chyb a nesíťové části byly zastaralé už v době uvedení. Zpětné ohlasy od uživatelů, ale velmi pomohly při dalším vývoji.

V září 1982 oznámil Bill Joy, že odchází k Sun Microsystems. Práce na začleňování TCP/IP stacku převzal Sam Leffler. Přesto se podařilo dodržet termín stanovený DARPA a 4.2 BSD vydat. Popularita nové verze byla ohromující,

během několika měsíců se prodalo přes tisíc licencí, víc než za celou dosavadní historii. Většina dodavatelů hardware začala 4.2 BSD nabízet místo komerčního System V od AT&T, protože ten neměl podporu sítí.

4.3 BSD

Přestože byla 4.2 BSD revoluční, setkala se také s kritikou. Uživatelé nebyli spokojeni s rychlostí systému a odladěností nových vlastností. Po dvou letech ladění a úpravách síťového kódu měla být proto vydána nová verze.

Vydání nové verze se ale zkomplikovalo. Firma BBN správně upozorňovala, že Berkeley nikdy do BSD nezačlenila finální verzi jejich implementace TCP/IP. Právě firmu BBN totiž původně DARPA vybrala pro vytvoření implementace. V novém vydání BSD by tedy měl být právě jejich síťový kód.

V 4.2 BSD se používala implementace napsaná Joyem a založená na původním prototypu od BBN. Na žádost DARPA porovnal Mike Karels z Berkeley obě implementace. BSD verze byla lepší, ale implementace od BBN byla novější podporovala některé nové vlastnosti, které v původní předběžné verzi TCP/IP nebyli. Mike Karels rozhodl dále používat implementaci z Berkeley a převzít nové věci z BBN implementace. BBN se ale nedala a jako kompromis navrhla dát do nové verze obsahovat obě implementace a nechat uživatele, aby se pro některou z nich rozhodli.

Kompromisní návrh na dvě implementace se nelíbil DARPA. Udržování dvou implementací by bylo finančně náročné a mohlo by vést k pozdějším problémům. Proto DARPA provedla na obou implementacích testování. Během testů se ukázalo že kód z Berkeley je efektivnější, ale BBN kód si vede lépe při vysokých zatíženích. U testů nakonec rozhodlo to, že BSD kód bezvadně procházel všemi

testy a BBN kód se za určitých situací zbláznil. BSD implementace byla prostě již odzkoušená a stabilnější. Do 4.3 BSD se tedy nakonec dostala BSD implementace.

První 4.3 BSD byla uvolněna v červenci 1986. Opravovala řadu problémů a podstatně zvyšovala výkon. Hlavní změna v síťové vrstvě byla podpora DNS. Přesto nedokázala zabránit postupnému přechodu uživatelů zpět k System V, ten již převzal z BSD implementaci TCP/IP a proti BSD byl vyspělejší.

Networking Release 1

Pokud chtěl v osmdesátých letech někdo používat BSD, musel mít zároveň licenci na zdrojové kódy od AT&T. BSD totiž byla založena na Unixu od AT&T a protože BSD distribuce obsahovala i zdrojové kódy, potřeboval uživatel i licenci na tyto kódy. Poplatky za licence nevadily pokud byly nízké, AT&T se ale rozhodlo zvednout cenu z 99 dolarů na několik tisíc dolarů. Berkeley se proto rozhodla od AT&T odpoutat. Části kódu pocházející od AT&T měli být nahrazeny.

Audit zdrojových kódů byl záležitostí několika let. O TCP/IP implementaci z BSD ale projevily zájem další výrobci, hlavně IBM pro novou platformu PC. Proto IBM požádala Berkeley, aby vydala síťový kód vydala samostatně bez AT&T licence. Tento kód prokazatelně nepocházel od AT&T a Berkeley o něm proto mohla rozhodovat sama. Kód implementace TCP/IP a síťové utility z BSD tedy byly vydány samostatně v červenci 1989 pod názvem Networking Release 1.

Tento release byl revoluční. Kód byl kompletně pod open source licencí, která umožňoval šíření, modifikace a přebírání kódů. Přestože za poslání pásky Berkeley účtovala 1000 dolarů, bylo možné kód dále kopírovat bez poplatků. Jediná omezující podmínka bylo uvedení Berkeley jako původního autora.

Mnoho univerzit proto tento kód umístilo na veřejně přístupné FTP servery. Kód z Networking Release 1 byl velmi snadno dostupný, kvalitní a proto jej najdeme snad ve všech současných operačních systémech.

Networking Release 2 a pokračovatelé

Po dokončení auditu kódu a nahrazení všech částí od AT&T následoval jejich vydání v roce 1991. Networking Release 2 nebyl ještě zcela funkční operační systém. Jednalo se spíše o release zdrojových kódů pro výrobce hardware. Některé části nebyly zcela dokončeny.

Bill Jolitz zvládl během šesti měsíců dopsat chybějící části a udělat funkční systém. První zcela opensource BSD byla 386/BSD určená pro 386 procesory. Binárky a zdrojové kódy nebyly distribuovány na páskách, ale přes FTP servery. Jolitz si časem našel zaměstnání a o 386/BSD se přestal starat. Projekt převzala komunita a pod názvem NetBSD žije dál. Od NetBSD se později oddělila OpenBSD specializovaná na bezpečnost.

Pro komerční využití BSD byla založena společnost Berkeley Software Design, Incorporated (BSDI). Krátce po svém založení převzala kód Networking Release 2, dopsala chybějící soubory a začala je komerčně distribuovat. To zahájilo několik soudních sporů, protože nebylo zcela jasné odkud pochází zdrojový kód. BSDI také v reklamní kampani použila název Unix, který byl předmětem ochranné známky.

Spory se vedly hlavně mezi BSDI a Unix System Laboratories (USL), který po rozdělení AT&T vlastnila její práva k Unixu. Po několika letech sporu Novell koupil USL a spory ukončil. BSDI musela přestat používat název Unix. Z 18000 souborů tvořících Networking Release 2 musely být 3 odstraněny a do 70

souborů byl přidán copyright UCL. Přestože soudní spory zpozdily práci na BSD, staly se požehnáním. Konečnou dohodou bylo možná distribuce celého systému BSD pod opensource licencí.

4.4 BSD vyšla v červnu 1994. Měla dvě verze, 4.4BSD-Encumbered byla komerční a obsahovala části, které stále vyžadovaly poplatky za licence. Oproti tomu 4.4BSD-Lite byla zcela bez poplatků a byla distribuována pod licencí BSD. Kód byl volně přístupný a každý jen mohl začlenit do svého systému bez poplatků. Jediným omezením je uvedení odkazu na původního autora. Díky volné licenci najdeme kód z 4.4BSD-Lite snad ve všech dnešních operačních systémech. Z 4.4BSD-Lite vychází NetBSD, OpenBSD a FreeBSD. Od nich ve velké míře přebíral kód Linus Torvalds pro svůj Linux, Microsoft pro Windows a Apple pro svůj MacOSX. Bez nadsázky lze říct, že původní BSD TCP/IP implementace se stala základem internetu.

Kame projekt

Kame projekt[3] měl za úkol vytvořit implementaci IPv6 a IPsec do systémů FreeBSD, NetBSD a OpenBSD. Projekt byl zahájen v roce 1998 a úspěšně skončen v březnu 2006. Projekt je společná akce japonských společností a univerzit. Japonsko zde nefiguruje náhodně, hlavně tato země má životní zájem na prosazení protokolů IPv6. Při rozdělování IPv4 nedostala zdaleka tolik adres kolik potřebuje. V této zemi probíhá obrovský nárůst mobilních telefonů a každý telefon znamená další IP adresu. Na projektu se podílely tyto společnosti a univerzity: ALAXALA Networks Corporation, Fujitsu Ltd., Hitachi Ltd., Internet Initiative Japan Inc., Keio University, NEC Corporation, University of Tokyo, Toshiba Corporation a Yokogawa Electric Corporation.

Projekt vnikl protože existovalo množství BSD derivací s víceméně stejným IP stackem. V roce 1998 začal být kladen důraz na IPv6, pokud by každá odnož BSD implementovala IPv6 samostatně, vedlo by to k velkému rozštěpení vývoje. Náklady na vývoj by neúnosně stouply. Proto byl zahájen společný projekt implementace IPv6. Tato implementace byla první použitelná a je v podstatě referenční. Výsledek tohoto projektu se dostal do FreeBSD, OpenBSD, NetBSD a přes ně i do MacOSX a pravděpodobně i do Windows.

Předchůdcem projektu byl WIDE Hydrangea IPv6/IPsec stack. Tato implementace byla výzkumná a byla ukončena po zahájení Kame projektu. Podle různých pramenů byla první implementací IPv6 vůbec. Bohužel jsem o ní nesehnal podrobnější informace.

Projekt Kame byl úspěšně dokončen. Jeho výsledky byly už v průběhu vývoje začleněny do BSD systémů a dále přebírány. FreeBSD používá kód od verze 4.0, NetBSD od verze 1.5. OpenBSD převzala pouze IPv6 implementaci, podporu IPsecu používá vlastní.

Projekt Kame přidává do kernelu tyto vlastnosti:

- rychlou robustní a stabilní podporu IPv6
- Kompletní podporu IPsecu pro IPv4 a IPv6.
- Podporu pro kompresi paketů IPComp
- podporu pro routování mezi IPv4 a IPv6, také zapouzdření IPv6 paketů do IPv4
- IPv6 discovery

V rámci projektu Kame byly vyvinuty také nezbytné programy pro práci s IPv6 sítí. Jsou to:

- Základní IPv6 sítové utility jako ping6, tracert6..
- DNS resolver
- DHCPv6 server a klient
- routovací programy pro IPv6

Cíle projektu bylo dosaženo a proto byl ukončen v březnu 2006 . Další výzkumné práce a vývoj přebírá projekt WIDE. Během osmi let vývoje se podařilo vytvořit robustní IPv6 implementaci. Tato implementace převzalo mnoho operačních systémů a protokol IPv6 je díky projektu široce podporovaný. Projekt tedy skončil úspěšně.

Implementace ve FreeBSD

Původní Berkeley Software Distribution (BSD) se rozštěpil do třech projektů: OpenBSD specializovanou na bezpečnost, NetBSD zaměřenou na portabilitu a FreeBSD. Tento systém je z derivátů BSD nejméně konzervativní a vývoj u něj probíhá nejrychleji. FreeBSD je největším konkurentem Linuxu. Je také portována na množství procesorových architektur.

FreeBSD je šířena pod licencí BSD, zdrojové kódy jsou tedy přístupné bez jakéhokoliv omezení. Na rozdíl od Linuxu s GPL licencí, může komerční firma kódy převzít a modifikovat bez jejich dalšího zveřejňování. Toho využívá řada firem, které z FreeBSD přebírají velké části kódu. Například implementace TCP/IP v MacOS X společně s mnoha dalšími částmi byla převzata z FreeBSD.

FreeBSD je vyvíjena jako kompletní operační systém se základními utilitami. Pod přímou správou je tedy nejen jádro, ale i systémové knihovny, základní utility, shell... Tímto se liší od GNU projektu, kde je každá část vyvíjena nezávisle na dalších a následně složené dohromady v distribuci. Zde je vývojář a distributor stejný.

FreeBSD je vyvíjena jako robustní a bezpečný server. Ve srovnání s GNU Linuxem a Windows je bezpečnější. Server Netcraft vypracovává seznam webových serverů s nejdelším uptime, zde má FreeBSD velmi slušnou pozici.

Historie

Vývoj FreeBSD [9],[10] začal v roce 1993 na základě zdrojových kódů 386BSD. Díky pochybnostem o původu zdrojových kódů a právním problémům bylo třeba původní zdrojové kódy zahodit a začít znovu.

FreeBSD 2.0 byla přepsána na základě zdrojových kódů 4.4BSD-Lite. Tato verze byla vydána v roce 1995. Objevuje se zde Mach Virtual Memory z CMU, který zajišťoval vysokou rychlost i při vysoké zátěži. Další novinkou byl systém softwarových portů, soubor skriptů, umožňující stahovat, kompilovat a instalovat software třetích stran. Systém portů se používá do dnes. Již v této verzi se FreeBSD ukázala velmi výkonná na síti. Proto se rychle rozšířila a začala sloužit v projektech jako Hotmail nebo Yahoo.

FreeBSD řady 3.x se nesetkala s takovým ohlasem. Přidala podporu pro víceprocesorové systémy, podporu ELF binárek a další vylepšení, tyto změny ale byly kritizovány jako nepodstatné a zpomalující systém. Hlavní výhoda FreeBSD, vysoká síťová rychlost, byla ohrožena.

Řada 4.x byla proto zaměřena hlavně na zlepšení výkonu. Povedlo se a FreeBSD se opět vrátila na špičku. Od verze 4.0 přebírá FreeBSD kód pro podporu IPv6 a IPsec z projektu Kame. Tato verze se široce používá dodnes. Změny, které zavedla další řada, způsobily mezi vývojáři spor. Od FreeBSD se oddělila DragonFlyBSD, založena na FreeBSD řady 4.

Současná stabilní řada FreeBSD je 5.x. Hlavní změny jsou přepsání systému paměti a systému nízkoúrovňových zámků, aby lépe vyhovovaly požadavkům víceprocesorových systémů. Další změny se týkají filesystému, FreeBSD implementovala ACL (Access Control Lists). Velmi zajímavá je také možnost zálohování celého disku k danému okamžiku, bez odstavení celého systému.

V současné době existují dvě vývojové řady 6.x a 7.x. Tyto by měli dále zlepšovat podporu víceprocesorových strojů [11]. Autoři také slibují vylepšit podporu bezdrátových sítí.

TCP/IP implementace

FreeBSD je stejně jako OpenBSD a NetBSD založena na 4.4BSD-Lite. Další velkou změnou bylo začlenění kódu z projektu Kame.

FreeBSD je zaměřena na výkonnost a efektivitu. Vývojáři FreeBSD chtějí z hardware vymačkat maximum a dosáhnout nejlepšího možného výkonu na daném hardware. Původní implementace z 4.4 BSD Lite byla proto značně modifikována.

Hlavní změnou je systém cachování síťových dat, tradiční cachování po stránkách bylo nahrazeno efektivnějším Universal Memory Alocatorem (UMA), který nemusí alokovat jednotlivé stránky po sobě, ale hromadně alokuje velké bloky paměti. Je kladen důraz na efektivitu u víceprocesorových strojů a s tím související mechanismus zámků (locking mechanism), ten byl dvakrát přepsán pro co nejlepší výkon. Nárůst výkonu po přidání nového procesoru je opravdu téměř lineární.

Další změny byly nutné, aby byl systém schopen obsluhovat co nejvíce TCP spojení současně. Několikrát byl přepsán mechanismus alokování připojení a udržování informací o něm.

Další změny doznal vnitřní filtr paketů a traffic shaping. Zcela se změnil mechanismus vyhodnocování a zahazování paketů. FreeBSD zavedla vlastní systém pro řízení rychlosti přenosu, ten si zapamatuje jakou rychlostí od něj klient stahoval data a při dalším spojení se vynechá stanovovaní optimální rychlosti a rovnou se začne s prověřenou rychlostí.

Implementace protokolů TCP/IP ve FreeBSD řady 6.x je v současné době pravděpodobně nejlepší ze všech.

NetBSD

Tento systém má nejblíže k původní 4.4 BSD-Lite. TCP/IP implementace se moc nezměnila. Vývojáři udělali několik změn pro zlepšení portability. NetBSD je systém podporující nejvíce hardwarových platforem na světě. Další změny spočívaly v přidání vlastností podle nových RFC. Ve verzi 1.5 byla přidána podpora IPv6 a IPSec z projektu Kame. TCP/IP implementace z NetBSD byla převzata v několika operačních systémech. Bohužel v současné době systém je vývoj strnulý a NetBSD zaostává za ostatními operačními systémy.

OpenBSD

Kvůli personálním sporům se od NetBSD oddělila OpenBSD. Její cíl je jednoduchý: maximální bezpečnost. Vývojáři OpenBSD jsou velmi dobří v kryptografii. Široce používané bezpečnostní nástroje OpenSSL i OpenSSH pocházejí právě z dílny OpenBSD.

Implementace TCP/IP v FreeBSD vybočuje po bezpečnostní stránce. Vývojáři jsou velmi opatrní v přidávání nových vlastností. Pokud se jim zdá některá nová vlastnost nebezpečná, raději ignorují RFC a nepřidají ji. Při přebírání nového kódu jsou stejně opatrní, pokud ho dokáží napsat bezpečněji, raději ho napíší znova. Například k IPSec implementaci z projektů Kame neměli důvěru a použily vlastní implementaci. Výchozí nastavení OpenBSD je paranoidní, všechno potencionálně nebezpečné je zakázáno. Také výchozí nastavení TCP/IP protokolu je velmi omezeno. Díky paranoidnímu nastavení je OpenBSD imunní proti všem útokům popsaných v této práci. Důraz na bezpečnost předurčuje OpenBSD jako router, firewall nebo zabezpečený server.

QNX

Operační systém QNX [6] sem zahrnuji pouze pro zajímavost. QNX je realtimový a mikrojádrový operační systém, malé jádro neobsahuje žádné drivery, pouze se stará o multitasking a správu paměti. Celé jádro je opravdu miniaturní, zabírá pouze 7 kB paměti. Ovladače hardwaru a implementace síťových protokolů běží jako obyčejné procesy mimo jádro. QNX má celkem tři implementace TCP/IP. První je starší implementace převzatá z NetBSD, podporuje všechny důležitá RFC, kromě IPv6. Druhá implementace je novější, pochází také z NetBSD, oproti starší verzi má začleněné výsledky projektu Cane a podporuje proto IPv6. Poslední je miniaturní implementace určená pro systémy s nedostatkem paměti zabírající pouze 80 KB.

Implementace TCP/IP v Linuxu

Historie

Linux[7] byl od začátku vyvíjen lidmi kolem celého světa, kteří spolupracovali po síti. Měl proto podporu síťových protokolů už ve velmi raných verzích. Úplně první byla podpora UUCP pro kopírování dat. Na implementaci TCP/IP se začalo pracovat krátce po vydání první verze. IP stack vytvořil na podzim roku 1992 Rosse Biro, ten je známa pod názvem Net-1. Druhou verzi Net-2 vytvořil v roce 1993 Fred van Kempen, po kompletním přepsání starší verze. Tato verze byla také první veřejně šířena, pod názvem Net-2d byla přidána do kernelu 0.99.10. Do Net-2d přispělo mnoho lidí, nejvýraznější z nich byl Alan Cox, jím modifikovaná verze Net-2Debugged se stala základem pro Net-3 a součástí kernelu 1.0. Net-3 zůstal také v kernelech řady 1.2 a 2.0. Před vydáním další řady byla síťová vrstva kompletně přepsána, součástí kernelu od řady 2.2 je tedy Net-4.

Detaily implementace TCP/IP v Linuxu

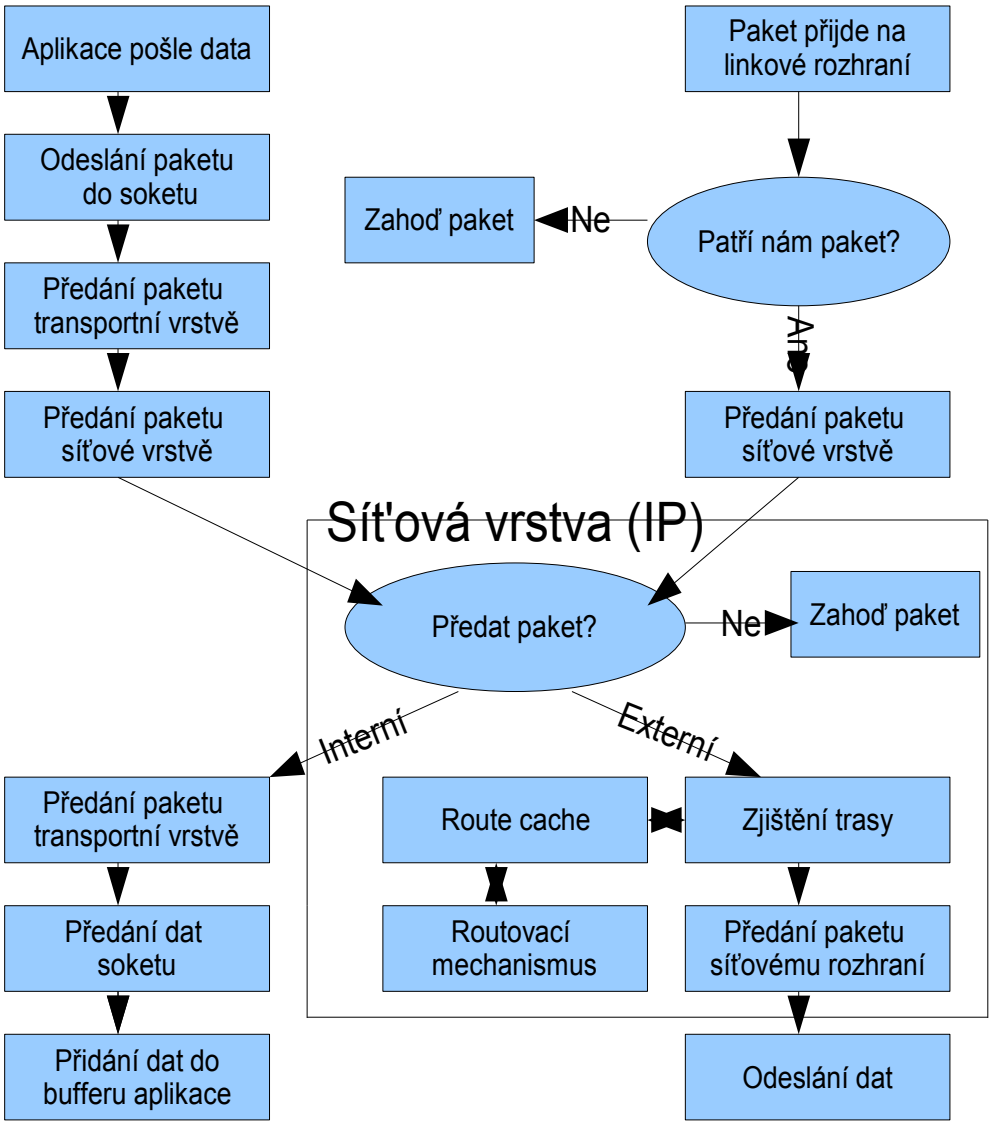


Diagram zobrazující zpracování IP paketů v Linuxu, autor Glenn Herrin [7]

V této kapitole by mohla vzniknout dvojznačnost ve výrazech, proto některé pojmy upřesním. Síťovou kartou se zde myslí linkové rozhraní, tedy rozhraní počítače komunikující po linkové vrstvě (ethernet karta, wifi karta). Další pojem je stack, je to technické označení implementace, IP stack je tedy implementace IP protokolu.

Informace pro tuto kapitolu jsem čerpal zejména ze zdrojových kódů linuxového kernelu [25]. Doplňující informace lze najít také v [22] a [23].

Implementace protokolu je závislá na konkrétním operačním systému, přesto zpravidla obsahuje tyto částí:

1. Kód pro přidání stacku do operačního systému. Obsahuje funkce pro start a ukončení stacku za chodu. Obsahuje také kód pro rozpoznání síťového protokolu.
2. Kód pro příjem paketu. Je volán při obsluze přerušení z linkové vrstvy.
3. Kód pro další práci s paketem, například filtrování, počítaní kontrolních součtů apod.
4. Kód pro odeslaní paketu. Je volán aplikačnívrstvou.
5. Kód pro routování paketu.

Jako konkrétní příklad jsem vybral implementaci TCP/IP verze 4 v Linuxu 2.4. (přesně 2.4.33.4) [25]. Její zdrojové kódy jsou volně přístupné a dobře zdokumentované. Tato implementace je několik let zmrazená (pouze se opravují chyby, nerozšiřuje se funkčnost) a její zdrojové kódy se proto příliš nemění.

TCP/IP stack v Linuxu je napsán v jazyku C a je multiplatformní. Zdrojové kódy[25] lze získat z adresy www.kernel.org společně se zbytkem linuxového kernelu, kód TCP/IP stacku se nachází v podadresáři **`net/ip4v`**. Samotná implementace má necelý megabajt. Další megabajt přidává Netfilter (**`net/ip4v/netfilter`**, filtr paketů) a IPVS (**`net/ip4v/ipvs`**, load balancing pro clustery), to ale do implementace přímo nepatří.

Příjem IP paketu

Síťová karta přijme linkový paket. Protože po linkové vrstvě může komunikovat více počítačů, je třeba vybrat pakety se správnou linkovou adresou (například MAC adresa u Ethernetu). Filtraci podle linkových adres provádí zpravidla samotná karta hardwarově, ale může ji provádět i ovladač síťové karty

softwarově. U novějších síťových karet (zejména WIFI) má síťová karta integrovaný vlastní mikropočítač a součástí ovladače je i firmware pro tento mikropočítač.

Pokud paket projde filtrací, síťová karta vyvolá hardwarové přerušení. Ovladač síťové karty zpracuje přerušení a přenese paket z bufferu síťové karty do hlavní paměti počítače. Pakety se zde řadí do dalšího bufferu a čekají na zpracování síťovou vrstvou. Ovladač síťové karty také informuje síťovou vrstvu o nových datech.

Operační systém zpravidla umožňuje používat více síťových protokolů nad jedním linkovým rozhraním. Proto v první linii ovladače síťové vrstvy je kód schopný rozpoznat síťový protokol (podle hlavičky paketu), který data dále předává vlastní implementaci síťového protokolu. V Linuxu je tento kód lokalizován v souborech **`net/core/dev.c`**

Část pro příjem paketů se nachází v souboru **`ip_input.c`**, konkrétně v metodě **`ip_rcv()`**. V této metodě se provádí různé kontroly (checksum, délka), pokud paket neprojde je zahozen. Dále je paket pro kontrolou předán netfiltru a vyhodnocen proti filtrovacím pravidlům. Posledním krokem je předání metodě **`ip_rcv_finish()`**.

Zde se rozhoduje kam paket předat dále. Jsou čtyři možnosti:

1. paket je určen pro tento počítač. V tomto případě se zavolá funkce **`ip_local_deliver()`** a paket je předán lokální transportní vrstvě.
2. Paket je určen pro cizí počítač. Je předán dále metodě **`ip_forward()`** v souboru **`ip_forward.c`**.
3. Paket je multicast určen pro více počítačů a vyžaduje speciální routování.
4. Nepodařilo se najít adresu, které paket předat. V tomto případě je vyvolána chyba a odeslán chybový ICMP paket.

Forwarding

O forwarding paketů se stará soubor **`ip_forward.c`** a funkce **`ip_forward()`**. První krok je ověření TTL (time to live) a jeho snížení o jednotku. Pokud číslo dosáhne nuly, je paket zahozen a stack odešle chybový ICMP paket. Na konci se paket předá funkci **`ip_forward_finish()`** a dále je odeslán metodou **`ip_finish_output()`** v souboru **`ip_output.c`**.

Odeslaní IP paketu

O odesílání IP paketů se stará soubor **`ip_output.c`** Paket nejdříve projde funkcí ip_output(), kde se případně upraví pro odesílání přes NAT. Následně se volá inline funkce **`__ip_finish_output()`**, zde se paketu nastaví číslo protokolu a předá se linkovému ovladači pro odeslání.

TCP implementace

Původně jsem čekal, že TCP protokol bude data získávat přímo od IP stacku. Mají k sobě blízko a zdrojové kódy jsou ve stejném adresáři. Ale funguje to jinak. TCP a IP stack jsou vzájemně odděleny a komunikují spolu pomocí asociovaných soketů. Funkce **`ip_local_deliver_finish()`** předává data ze síťové vrstvy transportní. Neděje se tak přímo, nejdříve se zkusí vyhledat soket asociovaný s transportním protokolem a číslem portu. Pokud není nalezen a transportní protokol je ochoten na daném portu ochoten první data (TCP vždy pro navazování spojení, UDP jen při otevřeném portu), vytvoří se nový asociovaný soket. Pokud byl soket v některém kroku nalezen, předají se mu data. V jiném případě se vygeneruje chyba a případně chybový ICMP paket. Kód obsluhující sokety je v souboru **`net/socket.c`**

Toto oddělení je logické. Transportní protokol TCP může komunikovat přes jakýkoliv síťový protokol. Lze si představit situaci, kdy IP pakety nahradí dopisy

a dráty pošťáci. Data by se přepisovala z dopisů přímo na vstup TCP stacku. Tento případ je sice úsměvný, ale přesto realizovatelný, dokazuje možnosti a dobrý návrh implementace. V linuxovém kernelu existuje jiný protokol přes který se TCP přenáší a to nová generace IPv6. Tento síťový protokol je zcela odlišný od IPv4, má jiné zdrojové kódy i adresář, přesto používá stejnou TCP implementaci jako IPv4

Úplné oddělení transportního protokolu od síťového by si vyžádalo snížení výkonu. Například optimální velikost paketů nebo stanovování rychlosti může u každého síťového protokolu probíhat jinak. V Linuxu je tento problém vyřešen elegantně. Pro oba síťové protokoly jsou zde „obaly“ v souborech **`net/ipv4/tcp_ipv4.c`** a **`net/ipv6/tcp_ipv6.c`**. Každý z nich obsahuje specifika daného protokolu a slouží jako most mezi IP a TCP protokolem. Obě implementace jsou ale plně odděleny, to lze snadno ověřit pomocí include direktivy. Jediné místo, kde se oba protokoly stýkají jsou právě tyto mosty.

První místo v TCP vrstvě kam přijde paket je funkce **`tcp_v4_rcv()`** v souboru **`tcp_ipv4.c`**. Zde se ověří kontrolní součet paketu a za použití funkcí v souboru **`tcp_input.c`** se soubor zařadí do fronty.

Zde je třeba vysvětlit rozdíl mezi asociovaným a transportním soketem. U bezstavového transportního protokolu (UDP) žádný rozdíl není. U stavového protokolu (TCP) se vytvoří asociovaný soket už při příjmu prvního SYN paketu. Transportní protokol je ale pořád ve stavu „navazuji spojení“. Navazování spojení (handshake) se provádí již přes asociovaný soket. Teprve po úvodní výměně paketů se transportní protokol přepne do stavu „spojení navázáno“, vytvoří transportní soket a data začne předávat aplikaci. Asociovaný soket se tedy vytváří pro každý přijatý paket z neznámé adresy, tedy i pro odmítnutá připojení. Toho zneužívá útok nazvaný SYN flooding, také popsaný v této práci.

TCP dělá samozřejmě i jiné věci než jen příjem a odesílání dat. Další velkou částí kódu je práce s transportními sokety. Tedy čekání na přípojení (listening), navazování připojení (handshake) a ukončovaní připojení. Kód pro obsluhu těchto stavů je v souboru `tcp.c`. Metody v tomto souboru pracují nad asociovaným soketem a přidávají k němu informaci o stavu v jakém se právě nachází TCP spojení. Existuje celkem deset stavů od „žádost pro připojení odeslána“ po „spojení ukončeno“.

Specifika Linuxu

Linux má velké množství vývojářů. Vývoj jádra je velmi rychlý a rozsáhlý. Linux vyniká hlavně množstvím podporovaných funkcí a protokolů. Navíc existuje množství neoficiálních patchů, které možnosti Linuxu dále rozšiřují. Linux je v současnosti nejflexibilnější operační systém.

Linuxová implementace není nejvýkonnější, v zátěžových TCP/IP testech dává lepší výsledky FreeBSD a Windows 2003. Linux má ale propracovanější multitasking a cachování, proto si při použití konkrétní aplikace (například http serveru) vede stejně dobře.

Další výhodou Linuxu je množství podporovaného hardware pro platformu PC. Z Unixových systému má bezpochyby největší množství ovladačů. Linuxová komunita je také dostatečně silná pro přesvědčení výrobců hardware k vydání hardwarových ovladačů pro Linux. Z poslední doby jmenujme například ovladače pro winmodemy nebo ovladač pro síťovou kartu u nForce čipsetů.

Nevýhodou Linuxu je určitá nestabilita vývoje. Současný kernel řady 2.6 je považován za vývojový a API se může změnit v každé podverzi. Starší a stabilní kernel řady 2.4, je sice roky prověřený, ale zase nepodporuje velké množství nového hardware a má citelně menší výkon. Není ale problém vybrat distribuci s dostatečně konzervativní politikou (Debian, Slackware) nebo garancí stability (RedHat nebo Novell Enterpise Linux).

Windows

Na začátku devadesátých let se Microsoft rozešel ve zlém s IBM a vývoj nového operačního systému se rozdělil na dvě větve. Na OS/2 u IBM a Windows NT u Microsoftu. Nový operační systém neměl v té době implementaci protokolu TCP/IP a neznal také sokety. Microsoft se rozhodl koupit zdrojový kód od firmy Spider Systems. Aby systém byl schopen využívat nové možnosti zavedl také Winsock API, které je součástí Windows dodnes. Součástí dodávky od Spider Systems byl také balík základních utilit, ty Microsoft přeportoval z BSD soketů na Winsock a přidal do systému. Kód od Spider Software byl založen na kódu z BSD, konkrétně Network Release 1.

Jako součást Windows NT 3.5 Microsoft uvedl zcela přepsanou implementaci TCP/IP. Tato implementace měla být revolučně výkonná a podporovat množství RFC. Implementace si bohužel výkonnostně nevedla dobře. Stack z Windows NT 3.5 Microsoft bez velkých změn použil i ve Windows 95, Windows NT 4, Windows 98, Windows SE a Windows Milenium [14].

Tato implementace TCP/IP od Microsoftu nebyla špatná. Podporovala základní RFC a byla dobře navržena. Špatná byla její integrace se systémem. Díky špatnému začlenění TCP/IP implementace do systému, nemohly být Windows použity při stejně vysoké zátěži, jako jiné systémy. Samostatnou kapitolou pak byly Windows 95 a Windows 98, kde byla implementace úmyslně ořezána, aby Windows NT neměly konkurenci. Tyto verze Windows tak například přišly o podporu více síťových adaptéru a forwardování paketů.

Windows 95 a 96 měly velké problémy s bezpečnostní TCP/IP implementace. Některé vážné chyby nebyly opraveny dodnes. To je ale nepřímo dáno také filozofií a návrhem těchto systému. Vážným nedostatkem byla nepřítomnost paketového filtru.

Přelomem jsou Windows 2000 [5]. Implementace TCP v tomto systému je velmi dobrá. Kromě podpory všech myslitelných RFC je také výkonná. Microsoft tvrdí, že TCP/IP implementace je založena na přepsaném stacku z Windows NT 3.5. Podle různých náznaků, ale převzal část síťového kódu z FreeBSD. Odpovídají tomu společné chyby v implementaci TCP/IP a FreeBSD je také uvedena v licenčních podmínkách Windows 2000. Také adresář **`C:\winnt\system32\drivers\etc`** obsahující síťové konfigurační soubory nápadně připomíná obdobné soubory na unixových operačních systémech.

Windows XP přidávají podporu IPv6 a několik aplikačních protokolů. TCP/IP implementace je stejná jako ve Windows 2000. Podporu pro IPv6 lze doinstalovat i do Windows 2000 a pomocí neoficiálního patche i do Windows NT 4.

Zajímavý je případ Service Packu 2 pro Windows XP [28]. Microsoft měl problémy s bezpečnostní Windows, mnoho virů zneužívalo síťovou službu RCP ke svému šíření. Tuto službu nelze zakázat, respektive lze, ale mnoho programů ji vyžaduje a tak bylo zvoleno „šalamounské“ řešení. Defaultně aktivovaný firewall pro všechna síťová připojení nepocházející z místní sítě. Ten je aktivován právě instalací Service Packu 2. Reakce na toto opatření jsou různé, faktem je že toto zabezpečení je účinné a virů viditelně ubylo. Díky špatné architektuře Windows lepší řešení není ani prakticky možné.

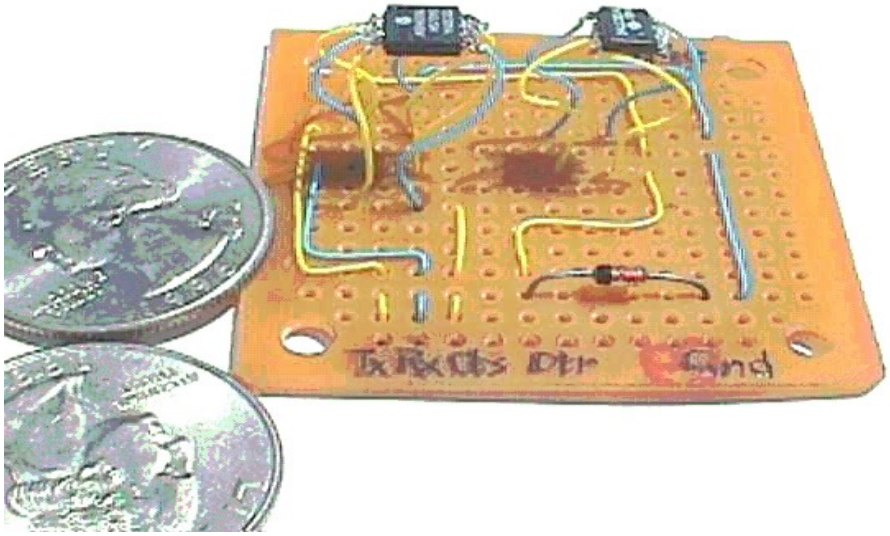
Málo známá je skutečnost, že SP2 také upravuje odesílání paketů. Microsoft se zřejmě snaží zabránit virům v masivním skenování sítě například pro odesílání spamu nebo dalším šíření virů. SP2 proto omezuje počet navazovaných spojení na neznámé adresy, počet současně navázaných spojení a celkový počet paketů odeslaných na cizí adresy. Negativní dopad má toto omezení zejména na uživatele P2P sítí, například Bittorrent.

Windows 2003 [4] představují další průlom, tentokrát výkonový. Windows v některých oblastech předstihují FreeBSD a Linux. Microsoft se také s modifikovanou verzí Windows se několikrát po sobě stává držitelem rekordu[30] v rychlosti přenášených dat. Díky vysoké integraci se systémem dosahují vyšší rychlosti také servery.

Windows Vista ještě nejsou na trhu. Podle tiskových zpráv by měla být síťová implementace zcela přepsána. Důvodem přepsání je lepší podpora víceprocesorových systému a IPv6. Implementace v současných testovacích verzích má ale vážné bezpečnostní problémy. Podle testování prováděném na testovací verzi [27] je zabezpečení žalostné. Popsané chyby jsou ale na novějších testovacích verzích jsou opravené. Možné chyby v TCP/IP implementacích jsou dobře popsané a Microsoft při testování bezpečnosti vychází pravděpodobně ze stejných zdrojů jako bezpečnostní analytici. Je jen škoda, že začal chyby v TCP/IP stacku opravovat těsně před uvedením operačního systému na trh.

Nejmenší TCP/IP implementace na světě.

Existuje několik implementací TCP/IP pro miniaturní programovatelné čipy PIC. Zpočátku jsem toto považoval za vtip, ale podle dalších zdrojů jsou tyto implementace opravdu skutečné. Všechny tyto implementace jsou součástí mánie o nejmenší web server, která proběhla v letech 1999 až 2001. Podporují pouze nezbytně nutnou část TCP/IP a pouze jedno TCP spojení současně. Jako hříčka jsou tedy dobré, ale v praxi nepoužitelné. Bohužel o nich nemám dostatek informací a proto je do přehledu zařazuji jen jako kuriozitu.



Fotografie iPic web-serveru. Autor Stephen Bannasch

Funkční TCP/IP implementaci lze umístit do velmi malého zařízení. Na přiloženém obrázku je nejmenší web server na světě. Využívá programovatelný čip PIC 12C509A běžící na 4 Mhz. Čip má paměť o velikosti 1024 slov (každé má 12bitů). Čip obsahuje minimalistickou implementaci TCP/IP namačkanou do 256 bytů, implementace zčásti podporuje IP, TCP, UDP a ICMP protokoly. Kromě toho na čipu běží HTTP 1.0 kompatibilní webserver a jednoduchý telnet server pro editaci souborů.

Implementace má být kompatibilní s RFC-1122, Host Requirements Document. Splňuje tedy všechny požadavky pro provozování jako veřejný web server. Web server byl skutečně v roce 1999 krátce veřejně přístupný, ale po několika měsících se odmlčel. Pak nejsou o projektu žádné nové zprávy.

Toto není jediný web server integrovaný do programovatelného PIC čipu. Tyto čipy jsou levné a rozšířené a tak pro ně existuje více implementací. Většinou podporují jen část TCP/IP protokolů nezbytně nutnou k fungování. Další implementace jsou:

WWWpic2 – implementace určená pro PIC 16F84. Je založena na předchozí implementaci. Autor dobře zdokumentoval stavbu hardware. Kód implementace je open-source. Neměl by tedy být problém stavbu zopakovat. Poslední aktualizace je z roku 2001. Více podrobností o této implementaci v [16]

WEBAce – WEB server založený na mikroprocesoru. Autor uvádí že k implementaci mu stačilo 1074 bytů. Také tento projekt není již několik let aktivní. Více podrobností o této implementaci v [17]

Webcard – web server běžící na čipové kreditní kartě. Karta má 6 KB EEPROM paměti a 1,2 kilobytů RAM. K dispozici jsou zdrojové kódy a technické návody. Projekt byl aktivní ještě v lednu 2006. Více podrobností o této implementaci v [18]

Útoky pomocí TCP/IP protokolů

Protokoly TCP/IP i přes skvělý návrh obsahují některé systémové chyby. Hlavní problémem se ukazuje možnost podvrhnout adresu odesílatele u IP paketů. Další problémem jsou chyby v jednotlivých implementacích. Obvyklý typ útoku je DOS (denial of service), který učiní cílový server nedostupný. Další typy útoků umožňují získat důležité informace o síti a počítačích. Hodí se proto pro průzkum sítě. Souhrnný přehled chyb a možných zabezpečení projednává [12]

ICMP

Zakázáním ICMP paketů můžeme zabránit zneužití našeho systému pro distribuovaný DOS útok. ICMP pakety také mohou prozradit hodně o našem systému, například informace o uptime. Možné útoky jsou:

- ICMP broadcast echo – útočník rozešle hromadnou žádost o odpověď s podvrženou zpáteční adresou. Odpovědi přijdou na jiný server a zahltí jeho linku.
- ICMP broadcast žádosti – útočník může zjistit podstatné informace o topologii naší sítě.
- ICMP změny routování – Toto je velmi závažná a podceňovaná chyba. Útočník může pozměnit naši routovací tabulku. Počítač se tak stane nedostupný. Útočník také může přesměrovat data přes svůj počítač a sledovat naše data.

Pro plné zneužití protokolu ICMP musí mít přístup do naší lokální sítě. Za firewallem nebo NATem není nebezpečí velké. Úplné zakázání ICMP paketů se není dobré, přestane totiž fungovat PING a TRACEROUTE. Obecně se proto doporučuje filtrovat ICMP pakety na firewallu a nechat je povolené na vnitřní síti a veřejně přístupných serverech.

TCP

Hlavní problém je v DOS útocích proti veřejným serverům. Těmto útokům se můžeme jen těžko bránit. Hlavní typ útoku je **SYN Flood** [13]. Spočívá v odeslání obrovského množství SYN paketů na otevřený port se žádostí o otevření připojení. Pakety většinou mají podvrženou adresu odesílatele, takže se zdá jako by útok byl prováděn z celého internetu. Server po obdržení SYN paketu musí odeslat zpět odpověď a dále v paměti držet informace pro pokračování připojení. Tímto útokem se tedy může vyčerpat paměť serveru nebo šířka pásma. Základní obrany proti SYN Floodingu jsou:

- použití TCP SYN Cookies - Tento způsob je jednoduchý a velmi účinný. Po přijetí SYN paketu server neotevírá připojení a neprovádí alokaci v paměti. Místo toho odešle zpět SYN/ACK paket se sekvenčním číslem vypočítaným jednoduchým algoritmem. Při přijetí ACK paketu se server podívá na sekvenční číslo a pokud vyhovuje jeho algoritmu, ví že s tímto připojením už pracoval. Pak teprve začne alokovat v paměti prostředky pro připojení. Tato obrana je zapnutá ve výchozím nastavení všech BSD, ve většině distribucích Linuxu a ve Windows 2003.
- Snížením povoleného počtu polootevřených připojení (HALF_OPEN TCP) – server se tedy stane nedostupným a DOS byl úspěšný, ale bude mít dostatek paměti. Správce se tedy bude moci na server připojit a provést obranu.
- Snížení časového limitu pro TCP handshake. Tímto způsobem významně odlehčíme serveru, ale zároveň vyřadíme klienty na pomalých linkách s velkými latencemi.
- Ignorováním prvních SYN paketů z neznámých adres. Toto je velmi nouzové řešení dobré hlavně pro pomalé linky. Pro běžné klienty to znamená pomalejší navázání připojení, jejich první SYN paket se ztratí a oni musí žádost zopakovat. Toto řešení by se mělo používat pouze pokud DOS útok již probíhá.

Idle scan

Idle scan [14] je anonymní oskenování portu cizího počítače. Útočník využívá chyby v implementaci TCP/IP u třetího počítače. Tzv zombie (zneužitý počítač) může být libovolný počítač připojený k internetu, není třeba na něj instalovat žádný program, stačí použít standardní vlastnosti TCP/IP.

Každý odeslaný paket má unikátní číslo fragmentace. Hodně starších operačních systémů toto číslo inkrementuje podle jednoduchého algoritmu. Ze znalosti dvou paketů (jejich čísel fragmentace) a operačního systému lze určit celkový počet odeslaných a přijatých paketů. Scan probíhá ve třech krocích. Útočník nejprve zjistí operační systém zombie a první fragmentační číslo. Poté útočník pošle žádost o připojení na cílový počítač s podvrženou adresou odesílatele. Cílový počítač odešle odpověď na zombie. V posledním kroku útočník získá útočník ze zombie počítače paket s druhým fragmentovým číslem. Z rozdílu fragmentových čísel lze poznat zda skenovaný port je otevřen, zavřen nebo filtrován.

DNS DOS útok

Toto je závažný problém [15], náš DNS server může posloužit jako zombie pro zprostředkování DOS útoku. Na malou žádost posílá DNS server několikanásobně delší odpověď. Při tomto DOS útoku tedy používá útočník několik veřejných DNS serverů k zesílení svého útoku. Útočník nejprve odešle na několik dotazů s podvrženou adresou oběti. DNS servery odpoví a data pošlou na adresu oběti. Pokud takto současně odpoví několik set serverů, spolehlivě přetíží síť oběti a učiní ji nedostupnou.

Proti tomuto útoku není spolehlivá obrana, klasickým řešením je omezit přístupy k DNS serveru na spolehlivé adresy. To ale dost dobře nelze, veřejně přístupný DNS server je podmínka pro provozování web serveru nebo mailu. Částečným řešením je omezení počtu dotazů na jednu IP adresu a po každém dotazu dát několika sekundovou penalizaci. Normální provoz nebude příliš narušen a útočník nebude moci náš DNS server příliš zneužívat.

Testování implementací

Bezpečnost TCP/IP implementací

Implementace protokolu TCP/IP je u každého počítače doslova v první linii. Útočník má k implementaci přímý přístup i v případě, že je počítač zabezpečen firewallem a není spuštěn žádný server. Implementace musí zabránit nejen pokusům o ovládnutí počítače, ale měla by být odolná i proti pokusům přetížit počítač nebo linku. Vážným bezpečnostním problémem může být i neschopnost rychle filtrovat pakety.

Je třeba odlišovat chyby v implementaci od chyb v aplikacích. Přímo součástí kernelu je jen relativně malý kód pro obsluhu transportních a síťových protokolů TCP, UDP, ICMP a IP. Ostatní protokoly spadající do rodiny TCP/IP jsou řešeny jako separátní programy běžící mimo jádro. Například klient protokolu DNS je v unixových systémech pouze několik funkcí v programové knihovně libc (například GNU libc). Podobná situace je ve Windows, kde DNS klient je také pouze soubor funkcí v programové knihovně.

Rozdělení na část v kernelu a programech je z hlediska bezpečnosti důležité. Program lze důsledněji zabezpečit, vypnout nebo nahradit za bezpečnější variantu. Pokud je nepostradatelný, lze firewallem odfiltrovat nebezpečné pakety. Navíc při kompromitaci útočník získá pouze práva programu. Při chybě přímo v jádru je situace horší. Kód běží v jádru a pokud ho útočník kompromituje (například pomocí buffer overrun) získá privilegia jádra. Na kompromitovaném počítači nelze věřit ničemu a jediné řešení je okamžité odstavení systému a obnova ze zálohy.

TCP/IP stack je relativně malý a má přesně danou funkčnost. Jeho bezpečnost je chápána jako kritická a proto bývá podroben velmi důkladnému testování. Prakticky všechny implementace jsou několik let staré a proto dobře otestované. Najít v současné době chybu v TCP/IP implementaci je proto velmi těžké. Například v [26] jsou za poslední tři roky evidovány pouze dvě vážné chyby přímo v TCP/IP implementaci ve všech sledovaných operačních systémech. Ostatní chyby jsou v programech nebo málo závažné.

Bezpečnost implementace TCP/IP u všech operačních systémů lze hodnotit jako velmi dobrou. Všechny operační systémy mají ve výchozí instalaci vypnuté potencionálně nebezpečná vlastnosti, například IP fragmentaci a různá druhy přesměrování. Ve všech OS jsou také implementovány bezpečnostní rozšíření například proti SYN floodingu. Prakticky všechny zde popisované útoky proti TCP/IP implementaci tak dnes nelze provést.

Přestože jsou implementace TCP/IP ve Windows velice dobré a bezpečné, doporučuje většina odborníků nepoužívat Windows na kritických místech sítě. Může za to nedobrá pověst Microsoftu z hlediska bezpečnosti a nelogické chování ohledně Windows XP SP2. Dalším podstatným důvodem je nemožnost provést nezávislý bezpečnostní audit zdrojových kódů TCP/IP implementace. Zdrojové kódy většiny ostatních OS jsou přístupné minimálně nezávislému auditorovi.

Největším bezpečnostním problémem implementace je odolnost vůči DOS útoku. Tomu nelze dost dobře zabránit, i nejlepší systém podlehne hrubé síle a početní převaze, ale lze minimalizovat její dopad. Hlavní obranou je rozumné nastavení implementace. Detaily obrany jsou diskutovány v kapitole o možných útocích.

Výkonnost TCP/IP implementací

Velkým problémem implementace protokolů TCP/IP je výkon. Deseti nebo sto megabitové Ethernety jsou tu desítky let. Přesto nebylo dlouho možné využít jejich plnou rychlost. Důvodem byl nedostatečný výkon počítačů a také neefektivní využití procesoru u starších implementací. Rychlost implementace se obvykle zlepšuje s každou novou verzí, jak programátoři opravují problémy a provádějí optimalizace. Je ironie že TCP/IP stacky nejefektivněji využívající CPU se objevují několik posledních let, v době kdy máme gigahertzové procesory.

Hodnotit výkonnost TCP/IP implementace je velmi problematické. Samotná TCP/IP implementace má několik desítek parametrů ovliňující její výkonnost. Existuje několik možných použití operačního systému a každé vyžaduje jiné nastavení hodnot pro optimální výkon. Výchozí hodnoty jsou obvykle zvoleny jako kompromis mezi několika možnými použitími. Například u desktopu je kladen důraz na maximální rychlost několika málo současných TCP spojení. Je třeba aby se soubor stáhl co nejdříve. U serveru se naopak vyskytuje hodně TCP spojení, každé z nich musí mít omezenou rychlost. Důvod je jednoduchý, rychlost linky a zdroje serveru jsou omezené a server potřebuje zabránit přetížení.

Proto se obvykle výkonnost TCP/IP implementace hodnotí podle dvou kritérií:

Maximální dosažená rychlost

Toto kritérium je typické pro desktop a fileservery s několika málo uživateli. Obvyklý problém těchto testů je, že neměří ve skutečnosti rychlost implementace, ale rychlost přístupu na pevný disk. V době gigabitových sítí je slabým místem počítače právě harddisk.

Existují dokonce oficiální rekord [30] pro dosažení maximální přenosové rychlosti. Tohoto rekordu se dosahuje v laboratorních podmínkách a s experimentálními TCP/IP stacky. Oba počítače znají parametry připojení a data dopředu, takže server si může celý soubor nahrát do paměti, zde data zabalit do paketů a po příchodu požadavku v podstatě vysypat na linkové rozhraní. Tyto rekordy ukazují spíše úroveň síťového hardware. S reálným nasazením nemají příliš společného.

Kvalita implementace není dnes již omezujícím faktorem pro dosažení maximální přenosové rychlosti. Nejdůležitější je zda jsou data uložena v paměti nebo na disku. O tom obvykle rozhoduje použitý serverový program. Dále do hry vstupuje řada nastavení operačního systému pro práci s pamětí a sítí. Za této situace je prakticky nemožné objektivní srovnání implementací z hlediska maximální dosažené rychlosti. Například podle studie [31] sponzorované Microsoftem vydané v roce 1999 byly Windows NT 4.0 několikanásobně rychlejší jako fileserver než Linux 2.2. Podle testu, který provedli fanoušci Linuxu [32], vyšel několikanásobně rychlejší naopak Linux.

Pokusil jsem se test maximální rychlosti zopakovat. Vzal jsem starý počítač (Pentium II 400 MHz, 128 MB Ram, disk Seagate Baracuda 32 GB 7200 otáček) jako server a druhý podstatně novější (Barton 1.9 GHz, 1024 MB Ram) jako klient. Druhý počítač je několikanásobně rychlejší, aby ze serveru vyždímal maximum a netestovala se schopnost klienta stahovat data.

Na server jsem postupně nainstaloval dva operační systémy: Windows 2000 Pro SP2 a Debian 3.1 s Linux kernelem 2.4.27. Jako server jsem použil osvědčený Apache HTTP Server verze 1.3.36. Na klientu byl nainstalován Arch Linux s jádrem 2.6 a stahovaní probíhalo programem wget. Testovací soubor byl archiv o

velikosti 500 MB, stahoval se z disku prvního počítače protokolem http na ramdisk ve druhém počítači. Použitá síť byl Ethernet 100 Mb.

V obou případech se data stahovaly stejnou rychlostí 3.5 MB/s. Pokusně jsem změřil sekvenční rychlost čtení z disku v prvním počítači, byla 3.5 MB/s. V jiném počítači je tento disk výrazně rychlejší, omezení rychlosti čtení asi způsobuje diskový řadič na desce. Trochu zajímavější výsledky ukazovalo vytížení procesoru. V obou případech divoce kolísalo, mezi nulou a 100%. Vytížení procesu httpd (web server) bylo prakticky konstantní, kolísání zátěže způsobovaly výpočty jádra. Nevím zda spojené se čtením dat z disku nebo TCP/IP implementací.

Jako druhý test jsem vzal 15 MB velký soubor a uložil ho na ramdisk v obou systémech. V obou případech se data stahovala maximální rychlostí možnou v 100 Mbps ehternetu.

Průkaznější výsledky by bylo možné zjistit při použití pomalejšího procesoru. Bohužel se mi nepodařilo sehnat pomalejší počítač ani podtaktovat stávající. Přesto tento malý test dokazuje, že v dnešních systémech není TCP/IP implementace hlavní překážkou pro dosahování vysoké rychlosti přenosu.

Maximální počet současně obsloužených TCP spojení.

Pro serverové nasazení operačního systému je důležitý maximální počet současně obsloužených spojení. První důvod je nutnost obsloužit současně co největší počet klientů, druhým důvodem je větší odolnost proti DOS útokům. Server má většinou maximální počet připojení omezen, aby se zamezilo jeho přetížení. Přesto ale existují aplikace které na limity OS mohou narazit.

Příklad serveru vyžadující velký počet připojení je například proxy server distribuující požadavky mezi další počítače v clusteru. Dalším příkladem mohou být různé P2P aplikace. Velká zátěž je také kladena na web servery poskytující drobný statický obsah, například obrázky pro web stránky, titulní stránky či nejrůznější počítadla.

Pro implementaci není těžké obsluhovat i několik desítek tisíc připojení současně. Je třeba jedině přiřadit paket asociovanému soketu. Tyto sokety jsou udržovány v poli s indexem podle čísla portu a vzdálené adresy. Vyhledávání v tomto poli je velmi rychlá operace. Slabým místem řetězu je obvykle návrh zbytku operačního systému a aplikace, protože ty musí velkou část dat držet v paměti a obsluhovat je. Také kvalita síťové linky může být slabým místem.

V praxi se nastavuje limitní počet připojení. Důvodem nejsou omezení v TCP/IP implementaci, ale v jiných částech počítače. Prvním je kapacita linky. I když je server připojen 100Mbps či rychlejší linkou, několik tisíc současných připojení by ji mohlo zahltit. Samozřejmě záleží na charakteru přenosu, při slabší provozu linka zvládne i maximální teoretický limit.

Druhým a podstatnějším omezením je kapacita počítače. U serveru zpravidla platí, že každé další připojení je obsluhováno ve zvláštním vlákně (thread). To klade obrovské požadavky na procesor a paměť počítače. Operační systém musí čas jednoho nebo několika procesorů spravedlivě rozdělit mezi všechny bežící vlákna. Přepnutí procesoru z jednoho vlákna na druhé není zcela triviální operace a zabere fixní čas. Při obrovském počtu vláken začne rychle klesat efektivita přepínání a procesor v podstatě neprovádí výpočty, ale většinu času se přepíná.

Další problém je kapacita paměti. Pro každé další vlákno je třeba zkopírovat část dat v společné paměti. Je třeba kopírovat všechna nestatická data, aby se

zabránilo současné modifikaci dat z několika vláken současně. To představuje problém hlavně u interpretovaných jazyků, protože je třeba zkopírovat i velkou část knihoven.

Toto omezení neplatí pro statická data, jako jsou soubory. Protože disk má relativně dlouho přístupovou dobu, je ale třeba mít všechna data v random access memory, typicky operační paměti počítače nebo flash paměti. Dalším omezením je kapacita linky. Jediná reálná aplikace, která je tedy schopna obsloužit i teoretický limit připojení je server poskytující drobné soubory. FTP protokol má příliš dlouhé navazování připojení a pro tuto aplikaci se příliš nehodí, takže zbývá HTTP protokol.

Pro tento protokol existuje množství serverů snažících se dosáhnout co největšího výkonu. Například existuje http server s názvem kHTTPd[33] integrovaný do Linuxového kernelu. Dále existuje množství minimalistických serverů (například Boa). Tyto servery ale jsou naprosto minimalistické a mohou posloužit pouze pro statické soubory. Pro provoz vytíženého http serveru je dobré oddělit statická a dynamická data. Statická data (obrázky, reklamy, titulní stránky) obsluhuje relativně pomalý a levný počítač s rychlou linkou a dostatkem paměti. Toto řešení je například použito pro server idnes.cz, který má obrázky na serveru i.idnes.cz.

Testy srovnávající maximální počet připojení TCP stacku prakticky neexistují. Většinou se jedná o testy srovnávající maximální počet TCP spojení pro program, například http server. Dobrý způsob testování je napsat si vlastní jednoduchý server a přeložit ho pro více operačních systémů.

Závěrečné srovnání TCP/IP implementací

Při vypracování přehledu jsem paradoxně došel k závěru že implementace TCP/IP v současných systémech *BSD, Linux a Windows jsou prakticky stejné. Jádro protokolů TCP/IP podporují všechny stejně dobře. Liší se pouze v množství podporovaných RFC. Všechny systémy mají dobrou podporu IPv6 a IPsec. Kvalitativně a výkonnostně mezi nimi není příliš velký rozdíl.

Jednotlivé implementace se od sebe liší především výchozím nastavením, které ovlivňuje jejich chování. Zde vyčnívá OpenBSD, která má téměř vše ve výchozí instalaci zakázané. FreeBSD má výchozí hodnoty nastaveny s ohledem na maximální výkon. U Linuxu se výchozí nastavení liší u každé distribuce. Windows mají také u každého verze jiné výchozí hodnoty, zejména podle určení (Server, Desktop)

Z bezpečnostního hlediska zcela jasně vede OpenBSD. Implementace TCP/IP je zde od začátku psaná s ohledem na bezpečnost a OpenBSD také těží z určité paranoii autorů k cizímu kódu. Bezpečnost ostatních implementací TCP/IP je také dobrá. Implementaci nelze nikde zneužít k průniku na počítač, všechny systémy jsou také zajištěné proti běžným DOS útokům.

Zcela jasně zaznamenatelné je postupné zlepšování TCP/IP implementace ve Windows. Ve Windows 2000 se implementace dramaticky zlepšilo množství podporovaných funkcí a bezpečnost. Windows 2003 Microsoft znamenají další velký krok v zabezpečení a výkonu. Tento systém se hravě vyrovná FreeBSD a Linuxu, v některých oblastech je dokonce předstihuje. Bude zajímavé jaké novinky přinese Vista a její případná serverová edice.

Množství podporovaných funkcí

Dlouhou dobu v množství podporovaných funkcí kralovala FreeBSD, implementace v tomto systému prakticky sloužila jako referenční. Její místo ale postupně přebírá Linux. Také Windows 2003 podporují prakticky všechny RFC doporučení. Z hlediska podporovaných funkcí není mezi touto trojicí velký rozdíl. V začleňování novinek je dnes nejrychlejší Linux.

OS	Linux 2.4	Linux 2.6	FreeBSD 6	OpenBSD 3.9	Windows 2000	Windows 2003
Dead gateway detection	Ano	Ano	Ano	Ano	Ano	Ano
Fast retranmit/recovery	Ano	Ano	Ano	Ano	Ano	Ano
Selective ACK (SACK)	Ano	Ano	Ano	Ano	Ano	Ano
Jumbo frame	Ano	Ano	Ano	Ano	Ano	Ano
Large windows	Ano	Ano	Ano	Ano	Ano	Ano
IP forwarding	Ano	Ano	Ano	Ano	Ano	Ano
NAT	Ano	Ano	Ano	Ano	Ano	Ano
IPSec	Ano	Ano	Ano	Ano	Ano	Ano
IPSec offload	Ne	Ano	Ano	Ano	Ano	Ano
Firewall	Ano	Ano	Ano	Ano	Ano	Ano
Packet Shapping	Ano	Ano	Ano	Ano	Ano	Ano
Block source routing	Ano	Ano	Ano	Ano	Ano	Ano
ICMP router discovery	Ano	Ano	Ano	Ano	Ano	Ano
IGMP v3	Ne	Ano	Ano	Ano	Ne	Ano
Checksum offload	Ano	Ano	Ano	Ano	Ne	Ano
Large send offload	Ne	Ano	Ano	Ne	Ne	Ano
DHCP	Ano	Ano	Ano	Ano	Ano	Ano
IPv6	Ano	Ano	Ano	Ano	Ne *	Ano
IPv6 IPSEC	Ne	Ano	Ano	Ano	Ne	Ano
IPComp	Ne	Ano	Ano	Ano	Ne	Ano
IPv6 discovery	Ano	Ano	Ano	Ano	Ne *	Ano
DHCPv6	Ano	Ano	Ano	Ano	Ne *	Ano

* vydaná později

Přehled podporovaných funkcí v OS

Zde je tabulka srovnávající podporované vlastnosti v jednotlivých operačních systémech. Velkou část TCP/IP protokolů podporují všechny operační systémy, proto jsem vybral méně obvyklé vlastnosti. U Linuxu jsem vycházel zejména ze zdrojových kódu kernelu [25], pro BSD systémy z vývojářských fór a Kame projektu[3], u Windows z oficiálních materiálů Microsoftu [4], [5] a [28].

Popis všech použitých vlastností je nad rámec této práce, proto stručně popíší jen jeden několik termínů.

Offload je „hardwarová akcelerace“ pro síťové přenosy. Operační systém pošle síťové kartě data a nechá je aby je odeslala. Na výsledek operace nečeká, proto termín offload. Nevýhodou je že operační systém se nedozví případný záporný výsledek operace a přenos nemůže případně zopakovat. Podmínkou pro offload je tedy kvalitní hardware a protokol schopný vyžádat si ztracená data (TCP). Všechny druhy offloadu jsou dílem Microsoftu, protože ten má dostatek prostředků aby ho prosadil mezi výrobci hardware. Opensource operační systémy implementují offload se zpožděním díky veřejně přístupné specifikaci.

IPv6 není třeba představovat. Do Windows 2000 lze částečnou podporu IPv6 doinstalovat, existují také ovladače pro Windows NT 4.

IPSec je rozšíření protokolu IPv6 o podporu šifrování přímo na síťové vrstvě. Tento standart byl zpětně „naroubován“ také na IPv4. Jeho implementace je náročná zejména na testování a bezpečnost.

Souhrn

Bezpečnost

Linux – horší, záleží na distribuci, velké množství hackerů

Windows – špatná, vážné chyby v aplikačním software, špatný návrh systému

FreeBSD – dobrá, pravidelné audity kódu, střední množství hackerů

OpenBSD – velmi dobra, bezpečnost je priorita vývoje, malé množství hackerů

Výkon

Linux – velmi dobrý

Windows – od verze 2003 velmi dobrý

FreeBSD – velmi dobrý

OpenBSD – dobrý, horší podpora víceprocesorů

Priority vývoje

Linux – vysoký výkon a množství podporovaných featur

Windows – podpora IPv6, efektivní využití víceprocesorů

FreeBSD – čistota kódu a přesná implementace RFC

OpenBSD – bezpečnost

Plány pro další vývoj

Linux – lepší podpora hardware hlavně bezdrátových adaptérů

Windows – zlepšení implementace ve Windows Vista

FreeBSD – větší efektivita na víceprocesorových strojích

OpenBSD – práce na vlastní implementaci IPSec, práce na bezpečné VPN

Doporučené použití

Linux – vše, univerzální

Windows – desktop, server pro intranet

FreeBSD – server, router, firewall

OpenBSD – router, firewall

Závěr

Implementace protokolů TCP/IP je velmi důležitá součást operačního systému. Všechny současné implementace vycházejí ze společných kořenů, starých přes dvacet let. Návrh implementace lze považovat za dobře zvládnuté odvětví softwarového inženýrství s dlouhou tradicí, propracovanou teorií a známými úskalími.

Vývoj se v současné době soustřeďuje zejména na lepší podporu IPv6 a efektivní využití víceprocesorů. V budoucnu lze očekávat překvapení díky nové TCP/IP implementaci ve Windows Vista a její případné serverové edici. Dalším překvapením může být výkon FreeBSD 7.

Práce podrobněji rozebírá implementaci v několika operačních systémech. Nejpodrobněji je popsána implementace v Linuxu 2.4, teoretické poznatky jsou zde demonstrovány na praktické implementaci v zdrojových kódech.

Všechny současné implementace (Linux 2.6, Windows 2003, FreeBSD 5...) jsou velmi dobré a navzájem prakticky stejné. Jejich srovnávání z hlediska výkonu nemá příliš smysl, protože implementace není slabým místem systému. Tím je obvykle rychlost pevného disku, nedostatek paměti nebo návrh aplikačního programu.

Také srovnání z hlediska podporovaných vlastností (featur) nemá velký smysl. Velká většina vlastností je podporována všude. Odlišné je zda je aktivovaná ve výchozí instalaci. Rozdílná může být také prověřenost implementace některé vlastnosti.

Všechny současné implementace jsou velmi bezpečné. Nelze je zneužít k ovládnutí systému. Nejvážnější chybou obvykle bývá možnost spustit DOS útok. Přehled bezpečnostních chyb eviduje pouze tři vážné bezpečnostní chyby v implementacích za poslední čtyři roky[26]. Bezpečnost implementace je velkou mírou dána výchozím nastavením.

Práce zpracovává přehled možných útoků prostřednictvím slabých míst v protokolech TCP/IP. Díky opatrně zvoleným výchozím hodnotám jsou popsané útoky v praxi nepoužitelné na všechny současné operační systémy. Slabým místem bezpečnosti systému je ale obvykle aplikační program, například RPC služba ve Windows XP.

V závěru práce je stručný souhrn rozdílů v jednotlivých implementacích. Společným problémem je podpora IPv6 a efektivní využití víceprocesorových strojů. Práce dále doporučuje ideální nasazení operačního systému z hlediska podpory TCP/IP.

V příloze je popsáno praktické využití znalostí TCP/IP implementací pro vzdálenou detekci operačního systému. Dále je popsána práce s programem NMap

Příloha: Detekce operačního systému přes síť

Jeden z prvních kroku před útokem je průzkum sítě. Útočník potřebuje znát uspořádání sítě a o použité počítače. Pokud má útočník informace o operačních systémech, přesně ví jaké chyby zneužít. Při útoku pak jde rychle a najisto. Jako součást zabezpečení serveru je tedy dobré zabránit detekci operačního systému.

Klasický způsob detekce

Tento způsob je nejjednodušší a zároveň nejpřesnější. Velká část serverů o sobě bez váhání prozradí operační systém, programy i přesná čísla verzí. Stačí se tedy připojit na server a počkat na uvítací zprávu.

```
artemis~> telnet  telnet.example.com
Trying 10.10.10.10 ...
Connected to telnet.example.com.
Escape character is '^]'.

RedHat Linux 7.1 (i386)

login:
```

Tento příklad ukazuje začátek telnetové session. Server běží na RedHat Linuxu verze 7.1. Podle čísla verze lze určit stáří a případné bezpečnostní chyby. Verze 7.1 pochází z roku 2001 a není několik let aktualizovaná. Případný útok má tedy velkou šanci uspět.

Jiný příklad ukazuje začátek http session. Mnoho http serverů přidává do hlavičky informace o svém verzi a použitých modulech. Opět z toho lze vyčíst důležité informace. Zde je http hlavička, kterou vrací jeden server:

```
HTTP/1.1 200 OK
```

```
Date: Mon, 05 Apr 2006 04:20:22 GMT
Server: Apache/2.0.54 (Debian GNU/Linux)
     mod_perl/1.999.21 Perl/v5.8.4
Cache-Control: private
Connection: close
Content-Type: text/html; charset=ISO-8859-2
```

Hlavička opět obsahuje přesná čísla verzí serveru včetně použitých modulů. Tento program nemá žádnou závažnou bezpečnostní chybu. Útočník proto asi ušetří čas a zkusí štěstí jinde.

Síťový otisk

Tento způsob používá síťový skener Nmap. Je založen na tom že každá implementace TCP/IP má svoje unikátní vlastnosti podle kterých ji lze detekovat. Program pro detekci provádí více testů a vylučovací metodou omezuje seznam kandidátů. S každým dalším testem je tedy určení operačního systému přesnější. Přesto tento způsob není zcela spolehlivý, výsledkem může být i více kandidátů nebo mohou testy zcela selhat. Základem je kvalitní databáze o chování operačních systému.

Dokument [19] popisuje různé metody detekce:

Fin test

Při tomto testu se zasílá FIN paket na otevřený port a čeká se na odpověď. Podle RFC 793 by neměla přijít žádná odpověď. Některé implementace ale odpovídají RST paketem. Týká se to HP/UX, MVS, IRIXu a Windows 95/98

Bogus flag test

Spočívá v nastavení některého z nedefinovaných TCP příznaků u SYN paketu. Zpět může přijít odpověď se stejně nastavenými příznaky. Tento problém se týká Linuxu před verzí 2.0.35. Některé operační systémy resetují spojení pokud tento packet dostanou.

TCP ISN čísla

Inicializační čísla TCP připojení (Initial Sequence Number) mají být podle specifikace náhodná. Výpočet náhodného čísla, ale může být náročný na procesorový čas. Proto mnoho implementací používá jednoduchý vzorec pro výpočet těchto čísel. Starší BSD inkrementovali číslo o konstantu. Novější systémy přidávají náhodné číslo určitém rozsahu (většinou 1-255) sem patří současné verze Solarisu, IRIXu, FreeBSD, Digital UNIXu a Windows od verze 2000. Starší verze Windows používají místo náhodného čísla systémové hodiny a inkrementují ISN podle aktuálního času. Poslední skupinou jsou systémy používají pseudonáhodných čísel, sem patří Linux, OpenVMS a AIX.

Znalost způsobu generování ISN čísla může mít také bezpečnostní dopad. Pokud jde předpovědět jedno nebo více inicializačních čísel, lze těmito čísly rozpojit existující připojení. Je tedy možné provést DOS útok na server zasláním několika RST paketů s těmito čísly. Je zde vysoká pravděpodobnost „trefit“ číslo již existujícího připojení a tím ho zavřít. Na straně serveru se zdá že klient korektně skončil připojení. Pro klienta je to naopak server kdo skončil připojení. Proti tomuto útoku není žádná obrana, jediné řešení je volit skutečně náhodné ISN.

IPID čísla

Podobný problém jako s ISN. Mnoho systémů inkrementuje IPID hodnotu s každým odeslaným paketem, například starší verze Windows zvyšují hodnotu IPID o 256.

TCP timestamp

Je další číslo, které může systém inkrementovat. Některé systémy tuto hodnotu nepodporují, jiné ji sice podporují, ale vrací nulu. Nejzajímavější je případ kdy systém tuto hodnotu inkrementuje v pravidelných intervalech (několikrát za sekundu), pokud není počáteční hodnota náhodná, lze z ní určit dobu od startu systému (uptime).

Příznak „nefragmentuj“

Tento příznak může zvýšit rychlost přenosu, proto může být u některých paketů nastaven. Velká část systému má ve výchozím nastavení fragmentaci zakázanou. Podle způsobu jakým systém fragmentaci obsluhuje lze určit operační systém.

TCP Initial Window

Jeho velikost je často unikátní pro daný systém, tehdy z ní lze velmi přesně určit operační systém.

ACK hodnota

Tento test spočívá v poslání sekvence FIN PSH a URG paketů na zavřený port. Většina implementací posílá zpět ACK hodnotu stejnou jako námi odeslané ISN Některé implementace, ale odesílají zpět náhodnou ACK hodnotu, nebo trochu pozměněné ISN.

ICMP Error Message Quenching

Podle RFC 1812 je dobré omezit vysílání chybových ICMP zpráv aby nedocházelo k přetížení sítě. Některé systémy se tímto doporučením řídí a mají nastavenou maximální rychlost pro odesílání ICMP paketů. Operační systém lze určit vyvoláním chybných stavů (například odesíláním UDP paketů na neotevřený port) a spočtením navrácených chybových ICMP paketů. Odesílání a příjem velkého množství paketů vyžaduje, ale více času a je třeba počítat se zahazováním ICMP paketů po cestě. Proto se tento způsob v praxi moc nepoužívá.

ICMP Message Quoting

Pokud router nemůže doručit paket, pošle zpět ICMP paket s chybovou zprávou. Podle RFC má být součástí tohoto paketu také část původního paketu. Většina implementací posílá pouze IP hlavičku + 8 dalších bytů. Některé systémy posílají více, konkrétně Solaris a Linux. Z toho lze opět částečně určit operační systém.

Další zdroj informací může být část paketu přeposlaná zpět. Protože paket už prošel určitým vyhodnocováním v systému je „částečně natrávený“. Některé implementace neudržují v paměti kontrolní součet původního paketu, ten se může tedy vrátit zpět deformovaný či úplně vynulován.

ICMP typ služby

Součástí ICMP paketů je hodnota TOS (type of service). Je dobrovolná a každý systém s ní zachází jinak. Například většina implementací používá vždy 0, ale Linux používá 0xC0.

Nepovinné TCP flagy

Jejich pomocí lze zjistit nejvíce. Jejich implementace je nepovinná, takže některé systémy je podporují a jiné ne. K otestování stačí odeslat jediný paket s dotazem a přijmout jeden paket s odpovědí. Proto se tento způsob velmi dobře hodí k rychlému průzkumu sítě. Bohužel přibývá operačních systému (FreeBSD 5, Windows 2000, Linux 2.6, MacOSX), které podporují všechny flagy. Tento způsob se tak omezuje na určování starších systémů.

Program nmap

Program nmap je síťový skener. S jeho pomocí lze zmapovat topologii sítě. Nmap dokáže také určit operační systém cílového počítače analýzou jeho síťového chování. Tento program je téměř kultovní, vyskytuje se například ve filmu Matrix II Reloaded. Jeho přínos pro síťovou bezpečnost a analýzu je nesporný.

Program je vyvíjen v Rusku . Nmap je šířen pod licencí GPL, to znamená že je bezplatně distribuovaný včetně zdrojových kódů, je open source. Program je multiplatformní, lze ho používat na většině operačních systému, včetně Linuxu, FreeBSD, Solaris, Windows a MacOSX. Některé operační systémy, ale nemají dost dobrou implementaci protokolu TCP/IP a proto na nich nelze využít všechny možnosti programu. Síťový provoz, který nmap generuje, není úplně standardní a proto je nutné ho pro plnou funkčnost spouštět pod privilegovaným uživatelem.

Program pracuje pouze v příkazovém řádku. Po spuštění program postupně skenuje cílové počítače a výsledek zobrazuje na standardní výstup. U každého skenovaného počítače zobrazí IP adresu, předpokládaný operační systém, uptime a seznam otevřených portů. K určení operačního systému a uptime nmap používá techniky popsané v dřívější kapitole.

Nmap byl původně pouze port scanner, později se z něj stal kompletní bezpečnostní nástroj. Zvládá několik technik skenování, které se liší odhalitelností a rychlostí. Nmap je také schopen obejít některé firewally a skenovat i přes ně.

Seznam použité literatury

[1] O'Reilly & Associates: Twenty Years of Berkeley Unix [online]

http://biblioweb.sindominio.net/telematica/open-sources-html/node22.html [30.5.2006]

[2] Wikipedia: Berkeley Software Distribution [online]

http://en.wikipedia.org/wiki/Berkeley_Software_Distribution [30.5.2006]

[3] Kame project: Kame overview [online]

http://www.kame.net/project-overview.html [30.5.2006]

[4] Microsoft corporation: Microsoft Windows Server 2003 TCP/IP Implementation Details [online]

http://www.microsoft.com/downloads/details.aspx?familyid=06c60bfe-4d37-4f50-8587-8b68d32fa6ee&displaylang=en [30.5.2006]

[5] Microsoft corporation: Microsoft Windows 2000 TCP/IP Implementation Details [online]

http://www.microsoft.com/technet/itsolutions/network/deploy/depovg/tcpip2k.mspx [30.5.2006]

[6] QNX Software Systems: QNX TCP/IP Networking [online]

http://www.qnx.com/developers/docs/momentics621_docs/neutrino/sys_arch/tcpip.html [30.5.2006]

[7] Glenn Herrin: Linux IP Networking [online]

http://www.cs.unh.edu/cnrg/gherrin/linux-net.html [30.5.2006]

[8] Bosko Milekic: Network Buffer Allocation in the FreeBSD Operating System [online]

http://bmilekic.unixdaemons.com/netbuf_bmilekic.pdf [30.5.2006]

[9] Wikipedia: Freebsd [online]

http://en.wikipedia.org/wiki/FreeBSD [30.5.2006]

[10] The FreeBSD Documentation Project: FreeBSD Handbook [online]

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/ [30.5.2006]

[11] Andre Oppermann: Optimizing the FreeBSD IP and TCP Stack [online]

http://people.freebsd.org/~andre/Optimizing%20the%20FreeBSD%20IP%20and%20TCP%20Stack.pdf [30.5.2006]

[12] Bob Cromwell: TCP/IP Stack Hardening [online]

http://www.cromwell-intl.com/SECURITY/security-stack-hardening.html [30.5.2006]

[13] Mariusz Burdach: Hardening the TCP/IP stack to SYN attacks [online]

http://www.securityfocus.com/infocus/1729 [30.5.2006]

[14] Fyodor: Idle Scanning and related IPID games [online]

http://www.insecure.org/nmap/idlescan.html [30.5.2006]

[15] CIAC: Domain Name System (DNS) Denial of Service (DoS) Attacks [online]

http://www.ciac.org/ciac/bulletins/j-063.shtml [30.5.2006]

[16] Vincent Sanders: WWWpic2 - A web server in a PIC [online]

http://www.kyllikki.org/hardware/wwwpic2/ [30.5.2006]

[17] Fedric White: webACE Server [online]

http://d116.com/ace/index.html [30.5.2006]

[18] University of Michigan: Webcard [online]

http://www.citi.umich.edu/projects/smartcard/webcard/ [30.5.2006]

[19] Fyodor: Remote OS detection via TCP/IP Stack FingerPrinting [online]

http://www.insecure.org/nmap/nmap-fingerprinting-article.html [30.5.2006]

[20] RFC 0793

http://www.cis.ohio-state.edu/rfc/rfc0793.txt

[21] RFC 1812

http://www.cis.ohio-state.edu/rfc/rfc1812.txt

[22] Harald Welte: The journey of a packet through the linux 2.4 network stack

http://ftp.gnumonks.org/pub/doc/packet-journey-2.4.html [5.12.2006]

[23] Oskar Andreasson: Ipsysctl tutorial

http://ipsysctl-tutorial.frozentux.net/ipsysctl-tutorial.html

[25] Linus Torvalds a další: Zdrojové kódy linuxového kernelu 2.4.33.4

Vydán 19.11.2006

http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.33.4.tar.gz

Zdrojové kódy **`net/ipv4/*`** a dále soubory s dokumentací

`Documentation/networking/tcp.txt` a

`Documentation/networking/ip-sysctl.txt`

[26] Secunia: vyhledávání klíčových slov TCP Stack [10.12.2006]

http://secunia.com/search/?search=TCP+Stack

[27] Tim Newsham a Jim Hoagland: Windows Vista Network Attack Surface

http://www.symantec.com/avcenter/reference/ATR-VistaAttackSurface.pdf

[28] Microsoft Corporation:

New Networking Features in Microsoft Windows XP Service Pack 2,

[15. září 2004]

http://www.microsoft.com/technet/community/columns/cableguy/cg0104.mspx

[29] Jeffrey B. Rothman, Ph.D. a John Buckma:

Which OS is fastest for High-Performance Network Applications?

[duben 2001]

http://john.redmood.com/osfastest.html

[30] Internet2 Land Speed Record

http://lsr.internet2.edu/

[31] Mindcraft: Web and File Server Comparison
[duben 1999]
http://www.mindcraft.com/whitepapers/first-nts4rhlinux.html

[32] Oliver Kaven: File Server Throughput and Response Times
[listopad 2001]
http://www.pcmag.com/article2/0,1895,17201,00.asp

[33] kHTTPd server [online]
http://www.fenrus.demon.nl/