VYSOKÁ ŠKOLA BÁŇSKÁ–TECHNICKÁ UNIVERZITA OSTRAVA

Fakulta elektrotechniky a informatiky

Moderní metody v počítačové a komunikační bezpečnosti pro integrovanou výuku VUT a VŠB-TUO

Garant předmětu:
Ivan Zelinka

Autor:
Ivan Zelinka

Ostrava 2014

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vznik těchto skript byl podpořen projektem č. CZ.1.07/2.2.00/28.0062
Evropského sociálního fondu a státním rozpočtem České republiky.

Za odbornou náplň tohoto vydání odpovídá autor. Ivan Zelinka je profesorem na Fakultě elektrotechniky a informatiky VŠB-Technické univerzity v Ostravě, kde přednáší předměty Bioinspirované výpočty a Počítačové viry a bezpečnost pro studenty navazujícího magisterského studia, kurzy jsou na fakultě nabízen ve studijním programu Informační a komunikační technologie.

Vznik skript byl podpořen projektem č. CZ.1.07/2.2.00/28.0062 Evropského sociálního fondu a státním rozpočtem České republiky.

Tato publikace neprošla redakční ani jazykovou úpravou.

Autor:	Ivan Zelinka
Katedra:	Katedra informatiky
Název:	Moderní metody v počítačové a komunikační bezpečnosti pro integrovanou výuku VUT a VŠB-TUO
Místo, rok, vydání:	Ostrava, 2014, 1. vydání
Počet stran:	138
Vydala:	Vysoká škola báňská-Technická univerzita Ostrava
Náklad	CD-ROM, 50 ks

Neprodejné

ISBN 978-80-248-3636-2

Motto: Nemůžeme se bránit proti útokům, kterým nerozumíme

Obsah

1 Úvod

Informace hrají v dnešní společnosti klíčovou roli. V počátečním období rozšiřování výpočetní techniky v minulých desetiletích byly na prvním místě náklady na pořízení, co největší dostupnost a také propojitelnost výpočetních systémů. Na zabezpečení většinou nikdo příliš nehleděl, protože jeho důsledná aplikace koneckonců IT řešení prodražuje a brání prvně jmenovaným cílům. Na přelomu tisíciletí ale došlo díky obrovskému počtu a rozmanitosti počítačových útoků a značným škodám s nimi spojenými k realistickému vystřízlivění a do popředí se nyní dostává především bezpečnost informací uložených v počítačích. Díky tomu dnes máme k dispozici spolehlivé antiviry, firewally, systémy detekce průniku, Windows Update, antispyware, spamové filtry, zdokonalené verze operačních systémů a další technologie, které nás chrání. Budou při takovém vývoji vůbec za několik let bezpečnostní specialisté potřeba? Podle názoru předních světových bezpečnostních expertů a podle našich každodenních zkušeností jednoznačně ano. Bezpečnosti se sice dostává nesrovnatelně více pozornosti než v minulosti, stejně tak ale roste počet a složitost technologií, které je nutné při ochraně brát v potaz. Dnes sice již pominula doba naivních útoků, kdy patnáctiletý hacker byl schopen spuštěním stažených skriptů shodit největší webové servery na Internetu (případ Mafiaboy, rok 2000), jistý si nicméně nemůže být opravdu nikdo. Stejně jako se zlepšila naše obrana totiž pokročily i technologie a postupy útočníků. Uveďme si příklady některých incidentů z roku 2008:

- chyba v systému DNS objevená Danem Kaminskim, která využívala tzv. narozeninového paradoxu k tomu, aby dokázala podvrhnout komunikaci s DNS serverem a otrávit jeho cache ohrožovala celou infrastrukturu Internetu tím, že útočníkům dovolovala přesměrovat libovolné webové a emailové servery. Pouze díky mimořádné spolupráci mnoha zainteresovaných subjektů po celém světě (výrobci DNS serverů, provozovatelé, ISP, ..) byly provedeny kroky pro odstranění chyby před její publikací na veřejnosti. Zda byla chyba objevena a zneužita ještě před jejím nalezením Danem Kaminskim není známo.
- první malware pronikl už i do kosmu – červ W32.Gammima.AG byl zachycen na mezinárodní vesmírné stanici IIS v notebooku jednoho z ruských kosmonautů.
- na letošním ročníku konference BlackHat byla pozornost soustředěna na možnost vývoje univerzálních rootkitů pro směrovače Cisco. Routery této značky se starají o zhruba 2/3 internetového provozu, pokud se vývoj takového rootkitu opravdu podaří mohou hackeři získat obrovskou moc.
- ve Francii hackeři ukradli peníze z bankovního účtu prezidentu Nicolasi Sarkozymu. Nejednalo se přitom o žádnou organizovanou skupinu na vysoké úrovni ale o dva drobné podvodníky. Není tedy příliš odvážné spekulovat, že proti cílenému cyber-útoku schopných útočníků by imunní málokdo.
- v Chile vystavil neznámý hacker ukradl a zveřejnil na Internetu osobní údaje šesti milionů Chilanů.
- bezpečnostní chyba byla objevena v systému nového Boingu 787, dolovala cestujícím přístup ze své veřejné sítě do sítě určené pro letové přístroje. Toto jistě další komentář nepotřebuje.

- zřejmě podstatná část moderních mobilních telefonů je zranitelných vůči útokům pomocí bluetooth nebo MMS. Například u některých verzí populární Motoroly RAZR k instalaci škodlivého software stačí pouze přijmout MMS s infikovaným JPEG souborem. Takový přístroj lze pak proměnit například v štěnici přeposílající veškeré hovory a textové zprávy. V budoucnu se také nejspíše dočkáme botnetů tvořených mobilními telefony
- bezpečnostní chyby byly objeveny v kávovarech značky Jura F90 (kávovar má síťové rozhraní umožňující servis na dálku). Samo o sobě to zní téměř komicky, nicméně úspěšné zneužití těchto chyb může vést až ke kompromitaci počítače s Windows XP spojeného s kávovarem. V současnosti jde spíše o zajímavou kuriozitu, ale není daleko doba, kdy téměř každé elektronické zařízení v domácnosti včetně elektrických zásuvek bude mít svou vlastní IPv6 adresu (koncept tzv. inteligentních domů). Napadnutelná plocha tedy opět poroste.
- V říjnu 2008 byla opět objevena chyba v RPC systému zasahující všechny v současnosti podporované verze operačního systému Windows (2000, XP, 2003, Vista, 2008). Pomocí chyby lze na vzdáleném systému spustit libovolný kód pod systémovým účtem (tzn. v nejhorším případě vzdáleně nainstalovat rootkit) . Chyba je natolik závažná, že Microsoft zcela mimořádně publikoval záplatu mimo běžný termín vydávání patchů (každé druhé úterý v měsíci). Tentokrát měl Microsoft štěstí protože chybu objevil dříve než druhá strana, jinak bychom se s největší pravděpodobností dočkali podobného červa jako byl nechvalně proslulý W32.Blaster v roce 2003.

Jak je vidět, počítačové útoky určitě neslábnou a fantazie jejich původců spíše rok od roku roste. Co víc – podle přední antivirové společnosti McAfee´s se zhruba od roku 2007 kultura internetového podsvětí velmi změnila v tom smyslu, že zmizeli mladí hackeři (tzv. script-kiddies), kteří vytvářeli viry a nabourávali počítače jen tak pro zábavu a ze zvědavosti a nahradily je organizované gangy profesionálních zločinců. Dnes v této oblasti jde hlavně o peníze. Vznikla celá šedá ekonomika, ve které je možné si služby hackerů, tvůrců spyware a pronajmutí botnetů nakoupit jako každý jiný produkt.

Hrozby tedy rostou a stejně tak musí růst naše znalosti, abychom se těmto hrozbám dokázali úspěšně postavit. V boji proti jakémukoliv protivníkovi je nezbytné znát svého nepřítele – jeho taktiku, dovednosti, nástroje a motivy. Činnosti, znalosti a dovednosti, které slouží k tomuto účelu se obvykle nazývají etický hacking, lidé kteří jej provádějí pak „white-hat“ hackeři. Nelze popřít, že informace, které jsou v této a podobných publikacích k dispozici lze stejně tak jako k prevenci zneužít k provádění počítačových útoků. To je ale nevyhnutelný aspekt vzdělávání a řešením určitě není takové informace utajovat a cenzurovat – druhá strana se k nim stejně dostane neboť i přes obrovskou rozmanitost motivů a úmyslů všechny hackery spojuje zvědavost a touha poznávat.

Firmy i jednotlivci musí vědět, jak k počítačovým škodám dochází, aby jim mohli předcházet. Účelem těchto skript je zvyšovat odbornost budoucích bezpečnostních profesionálů v boji proti zlovolnému hackingu. Cílem je zvýšit povědomí o problematice tak, aby byli v budoucnu schopni samostatně posoudit bezpečnostní rizika nejen svých počítačů ale také případně počítačů, které budou mít případně na starosti například jako správci sítě.

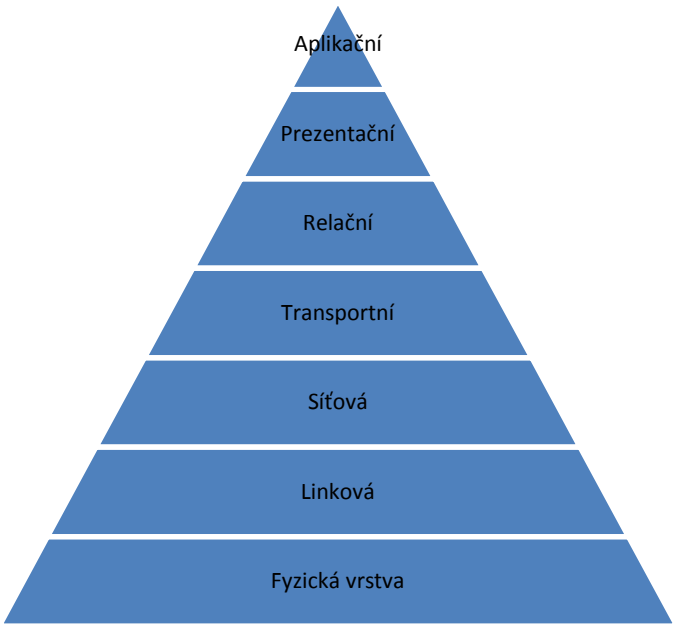
Veškeré čtenáře a studenty nicméně i tak nabádáme k tomu, aby si uvědomili dosah a vážnost následků pokud by získané informace zneužili v reálném prostředí.

2 Opakování základů počítačových sítí

V této úvodní kapitole si letmo zopakujeme základy počítačových sítí. Pro bezpečnostního specialistu je naprosto nezbytné, aby znal všechny aspekty síťové infrastruktury a velmi dobře ovládal nástroje pro práci s ní. Z hlediska navazujících kapitol je důležité především dokonalé pochopení a schopnost rozlišit zda se vzdálený systém ke kterému přistupujeme, nachází na stejné síti jako my nebo ne a jaké síťové prvky vstupují do hry ve spojení mezi námi.

2.1 Sedmivrstvý OSI model

Referenční model OSI vypracovala organizace ISO jako mezinárodní normu. Jeho úkolem je poskytnout základnu pro vypracování norem. Každá ze sedmi vrstev vykonává skupinu jasně definovaných funkcí potřebných pro komunikaci. Každá vrstva (kromě nejnižší) využívá služeb vrstvy pod sebou a své vrstvy poskytuje vyšší vrstvě (kromě aplikační). Pravidla komunikace mezi stejnými vrstvami různých systémů se nazývají protokoly.



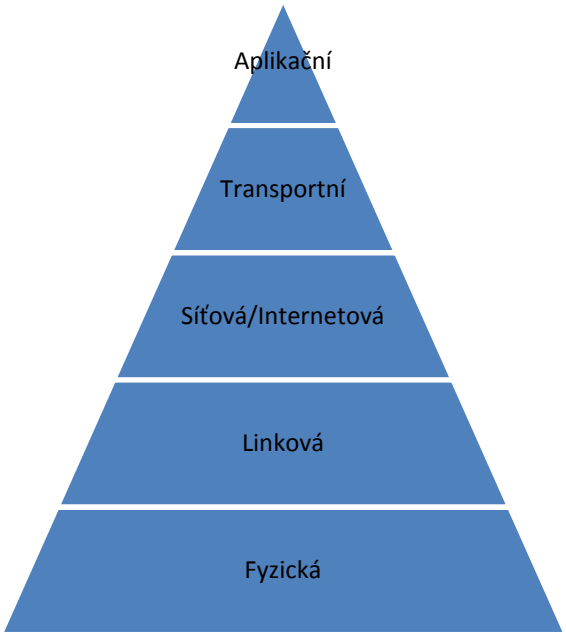
Obrázek 1: OSI model

1. **Fyzická vrstva:** definuje elektrické a fyzikální vlastnosti zařízení (rozložení pinů, tvary konektorů, napěťové úrovně, vlastnosti kabelů). Navazuje a ukončuje spojení s přenosovým médiem, stará se o konverzi signálů apod. Fyzické spojení může být dvoubodové (sériová linka) nebo vícebodové (Ethernet)
2. **Spojová vrstva:** poskytuje spojení mezi dvěma sousedními systémy. Jednotky dat se zde nazývají **rámce**. Seřazuje rámce, sleduje chyby ve spojení, nastavuje parametry linky. **Poskytuje spojení pouze mezi místně připojenými zařízeními.** Příkladem je Ethernet. Adresou zde jsou fyzické adresy zařízení – MAC (Media Access Control).
3. Síťová vrstva: stará se o **směrování** v síti a síťové adresování. **Poskytuje propojení mezi systémy, které spolu přímo nesousedí.** Jednotkou informace zde je paket. Nejznámějším protokolem této vrstvy je IP. Adresou jsou zde logické adresy zařízení – IP adresy.
4. **Transportní vrstva:** zaručuje přenos dat mezi koncovými uzly. Adresou pro tuto vrstvu je číslo portu na počítači. Jednotkou na této vrstvě je **segment**. Hlavními protokoly jsou TCP a UDP. TCP (Transmission Control Protocol) zajišťuje spolehlivý přenos dat. Používá se u těch aplikací kde je nepřijatelné aby se jakákoliv část dat po

cestě ztratila, což je většina běžných aplikací. Spolehlivost je zaručena potvrzováním paketů. Cenou za spolehlivost je nižší rychlost. UDP (User Datagram Protocol) nemá žádný mechanismus kontroly zda všechna data došly včas a jejich potvrzování. Data jsou prostě odeslána ze zdroje na cíl. Použití má tam kde je kritická rychlost a je možné tolerovat občasné výpadky dat – streamované video, internetové rádio, vyhledávání na DC++, online hry, DNS.

5. **Relační vrstva:** jejím smyslem je organizovat a synchronizovat dialog mezi spolupracujícími relačními vrstvami.
6. **Prezentační vrstva**: její funkcí je transformovat data do tvaru ve kterém jej požadují aplikace. Patří sem kódování, komprese, převod mezi různými kódy a abecedami, pořadí bytů apod. Patří sem například base64 kódování.
7. **Aplikační vrstva**: úroveň na které pracují příkazy jednotlivých aplikací: http, FTP, DNS, DHCP, POP3, TELNET, SMTP, SSH.

Pětivrstvý model TCP/IP



1. Aplikační vrstva

DHCP, DNS, FTP, Gopher, FTP, IMAP4, IRC, NNTP,POP3,SMTP,SSH, TELNET

2. Transportní vrstva

TCP, UDP, DCCP, SCTP, RTP

3. Síťová/Internetová vrstva

IP (IPv4, IPv6), ICMP, ARP, RARP, RIP, BGP, ICMPv6, IPsec

2. Linková vrstva

Ethernet, 802.11, 802.16, Wi-Fi, WiMAX, ATM, Token ring, Frame Relay, GPRS, PPP, ISDN

1. fyzická vrstva

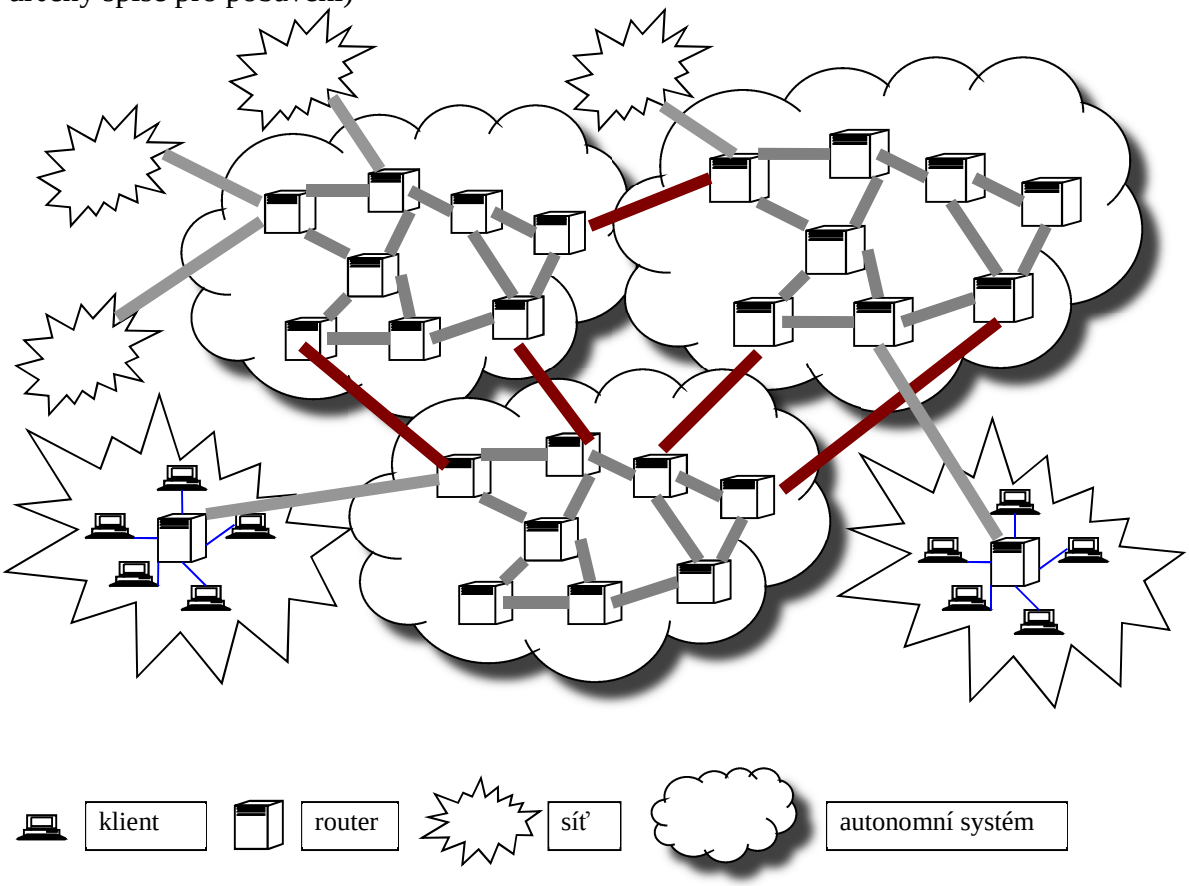
Fyzická vrstva Ethernetu, modemy, optické kabely, koaxiální kabely, kroucená dvoulinka

Obrázek 2: TCP/IP model

2.2 Internet

Internet je tvořen desítkami až stovkami tisíc různorodých sítí, které jsou navzájem propojeny pomocí homogenního protokolu IP, jenž umožňuje logicky adresovat požadavky z kteréhokoliv uzlu sítě do kteréhokoli jiného uzlu. Páteř Internetu tvoří navzájem propojené routery, které mezi sebou přesměrovávají IP packety. Na každý router napojeny buď jiné routery nebo přímo počítače pro něž router funguje jako brána k Internetu (ostatním sítím).

Počítače napojené na jeden router jsou navzájem v lokální síti a komunikují spolu pomocí místních hardwarových adres místo logických IP adres.
Vybrané údaje o rozsahu Internetu (počátek roku 2008):
Počet uživatelů Internetu: zhruba 1,2 miliardy
Počet aktivních www domén: okolo 100 000 000
Celková hmotnost elektronů přesunutých za jeden den v celém Internetu: 50 g (spekulativní údaj určený spíše pro pobavení)



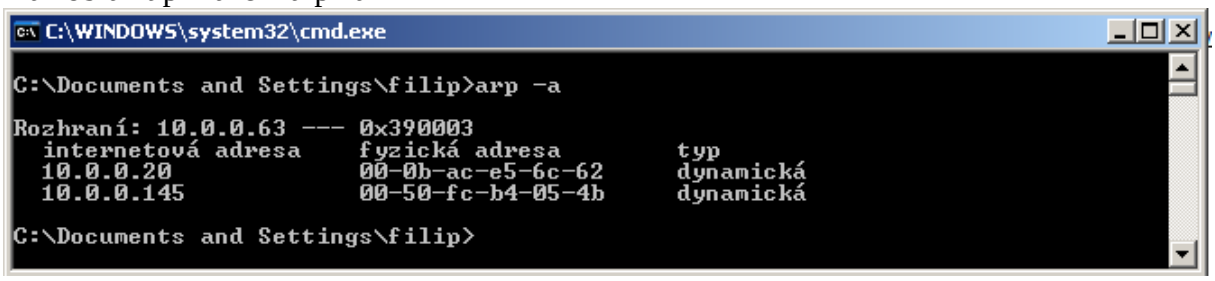
Obrázek 3: Schematická struktura Internetu

Klient je koncové zařízení. Každý klient na Internetu má svoji jedinečnou IP adresu. Nejčastěji se jedná o osobní počítač, může ale také jít např. o webové kamery, harddiskové rekordéry, měřící zařízení,..
Router je zařízení, které se stará o přesměrování dat správnými uzly sítě tak aby se ze zdrojové adresy dostaly do cíle. Jsou na něj napojeny jednotlivé koncové zařízení (tím tvoří místní síť) a ostatní routery.
Lokální síť je seskupení počítačů a síťových prvků (huby, switche, routery, access pointy) připojených k jedné bráně do Internetu (což je také router). Lokální síť je určena maskou IP adresy.
Autonomní systém je kolekce IP systémů a routerů pod správou jedné entity, která navenek zachovává jednotnou směrovací politiku. Každý autonomní systém má přidělené unikátní identifikační číslo ASN (původně 16-bitové, tzn. max 65000 AS, od roku 2007 32-bitové). Toto číslo je využíváno protokolem BGP (Border Gateway Protocol) pro směrování IP provozu mezi autonomními systémy.

2.2.1 MAC Adresa

V lokální síti (tzn. ve skupině počítačů, které jsou fyzicky za společnou branou do počítače) spolu počítače komunikují ne pomocí IP adres ale pomocí fyzických adres síťových karet.

Tyto adresy se nazývají MAC (Media Access Control) a každá síťová karta na světě má svou jedinečnou MAC adresu hardwarově „vypálenu“ již od výrobce. Tvoří ji 48 bitů, neboli šest bytů, které se nejčastěji zapisují jako šest číslic v šestnáctkové soustavě oddělené dvojtečkami nebo pomlčkami. První tři bajty určují výrobce, druhá trojice unikátní výrobní číslo karty.
V OS Windows lze MAC adresy počítačů na místní síti které operační systém v danou chvíli zná zobrazit příkazem arp –a



Obrázek 4: Příkaz ARP

2.2.2 Switch

Hlavní úkol switche (přepínače) je starat se rozesílání paketů mezi počítači na místní síti. Na switch jsou v hvězdicové topologii napojeny jednotlivé počítače. Každý počítač má svou hardwarovou MAC adresu. Pokud aplikace z jednoho počítače vyšle datagram aplikaci na jiném počítači je ale cílový počítač zadán IP adresou ne MAC adresou. Nejprve je nutné z IP adresy zdrojového počítače, IP adresy cílového počítače a tzv. masky sítě určit zda oba počítače leží na stejném segmentu sítě. Pokud ano, pak je nutné přeložit logickou IP adresu na hardwarovou MAC adresu. K tomu slouží protokol ARP (Adress Resolution Protocol). Po získání hardwarové adresy je packet vyslán do sítě. Switch si udržuje tabulku MAC adres počítačů, které jsou k němu připojeny. Pokud tedy od kteréhokoliv dojde packet, podívá se na cílovou MAC adresu, porovná ji se svou tabulkou a packet odešle pouze na ten port, který vede na cílovou MAC adresu. Jednotlivé packety tedy „přepíná“, podobně jako jsou přepínány telefonní hovory.

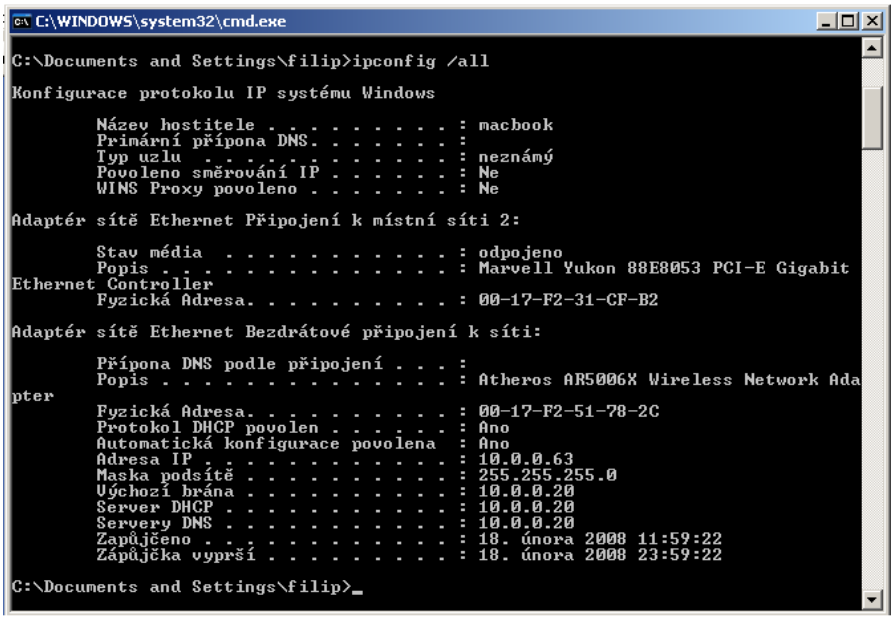
2.2.3 Hub

Hub (rozbočovač) je technologický předchůdce switche. Jeho úkolem je také rozesílat packety mezi počítači v místní síti které jsou na něj napojeny v síťové topologii. Rozdíl je v tom že pokud na hub přijde packet z jednoho počítače je tento packet poslán na všechny výstupní porty hubu bez ohledu na MAC adresu cílového počítače. Každý počítač si pak individuálně porovná MAC adresu uvedenou v packetu jako cíl se svou MAC adresou a pokud jsou stejné packet si načte, protože je určen pro něj, jinak jej zahodí. Nevýhodou hubů tedy je jednak menší propustnost sítě a jednak to že pokud je síťová karta na počítači který je k hubu připojen přepnuta do promiskuitního módu, může si přečíst veškerý provoz mezi všemi počítači v daném segmentu sítě a ne pouze packety určené jemu jako je to u přepínané sítě se switchem.

2.2.4 IP adresa

Současná verze IP protokolu se označuje IPv4. IP adresa je tvořena 4 oktety – bajty, které jsou nejčastěji reprezentovány jako čtyři desítkové čísla od 0 do 255 oddělené tečkou. Celkem adresu tvoří 32 bitů, 2^32 = něco přes 4 miliardy možných unikátních IP adres (pouze teoreticky, několik rozsahů IP adres je rezervovaných pro zvláštní účely) a tedy zhruba 4 miliardy současně zapojených síťových zařízení. Tento počet přestal již před několika lety stačit proto je navržen a do nejnovějších operačních systémů a síťového hardwaru již implementován nástupce označovaný jako IPv6. V tomto protokolu tvoří adresu 128 bitů. Úplné „přepnutí“ celého Internetu na IPv6 by si nicméně stále vyžádalo obrovské finanční náklady a konfigurační problémy proto je zatím v nedohlednu. Jako náhradní řešení je v současné době používána technologie NAT – Network Adress Translation kdy je pod jednu

IP adresu přidělena celá síť počítačů se soukromými IP adresami. Navenek všechny tyto počítače mají jakoby právě tu jednu veřejnou IP adresu. To je umožněno díky bráně která si udržuje tabulku soukromých IP adres „svých“ počítačů a každému požadavku který ze sítě odejde přiřadí jiný port na svém veřejném rozhraní.
IP adresu svého počítače zjistíte na OS Windows příkazem „ipconfig /all“ v příkazové řádce. Zjistit svou IP adresu na Internetu můžete například na stránce www.whatismyip.com. Porovnáním těchto dvou hodnot se můžete dozvědět například to zda máte veřejnou IP adresu (jsou stejné) nebo zda se nacházíte v síti ve které je zapnuta funkce NAT (Network Adress Translation) a máte tedy neveřejnou IP adresu (jsou rozdílné). IP adresy přiděluje správce sítě. Mohou být přiděleny buď staticky – při každém připojení do sítě má počítače stejnou IP adresu nebo dynamicky – po připojení do sítě je adresa přidělena protokolem DHCP.



Obrázek 5: Příkaz IPCONFIG

Kromě IP adresy (10.0.0.63) je zde také viděl MAC adresa (00-17-F2-51-78-2C), název hostitele (macbook), brána pro připojení do sítě (10.0.0.20) a tzv. maska podsítě.

2.2.5 IP maska podsítě

Maska podsítě slouží k určení toho, která část IP adresy určuje adresu sítě a která část určuje adresu počítače v síti. Funguje na principu logického součinu AND mezi maskou sítě a IP adresou. Ty bity v masce které jsou nastaveny na hodnotu 1 představují v IP adrese adresu sítě. Tato operace je velmi důležitá pro komunikaci mezi jednotlivými stanicemi na síti. Pokud totiž dvě stanice leží na stejné subsíti, komunikují spolu přímo v lokální (nejčastěji Ethernetové) síti pomocí fyzických MAC adres. Komunikace mezi nimi tedy nejde přes router (maximálně přes switch nebo hub). Pokud stanice leží každá v jiné síti, komunikují spolu pomocí logických IP adres a jejich vzájemná komunikace prochází přes router.

Příklad 1:
Stanice 178.89.168.1 pošle IP packet na stanici 178.89.178.5. Maska sítě obou stanic je nastavena na hodnotu 255.255.255.0. Leží stanice na stejné podsíti?

Stanice 1

178.89.168.1 = 10110010.01011001.10101000.00000001
&&

255.255.255.0 = 11111111.11111111.11111111.00000000
---------- síť 1
10110010.01011001.10101000.00000000 = 178.89.168.0

Stanice 2
178.89.178.5 = 10110010.01011001.10110010.00000101
&&
255.255.255.0 = 11111111.11111111.11111111.00000000
---------- síť 2
10110010.01011001.10110010.00000000 = 178.89.178.0

Jak je vidět adresy obou sítí jsou různé (178.89.168.0 ≠ 178.89.178.0), počítače tedy neleží ve stejné síti a nebudou spolu komunikovat pomocí fyzických MAC adres. Podle masky lze též odvodit, že pouze posledních 8 bitů IP adresy tvoří adresu počítače v síti – v této síti tedy může být až 255 stanic.

Příklad 2
Ležely by oba počítače z předchozího příkladu na stejné síti, pokud by maska sítě byla nastavena na 255.255.128.0?
Stanice 1
178.89.168.1 = 10110010.01011001.10101000.00000001
&&
255.255.128.0 = 11111111.11111111.10000000.00000000
---------- síť 1
10110010.01011001.10000000.00000000 = 178.89.128.0

Stanice 2
178.89.178.5 = 10110010.01011001.10110010.00000101
&&
255.255.128.0 = 11111111.11111111.10000000.00000000
---------- síť 2
10110010.01011001.10000000.00000000 = 178.89.128.0

V tomto případě jsou obě adresy sítí stejné (178.89.128.0 = 178.89.128.0) a oba počítače tedy leží na stejné podsíti. Budou spolu tedy komunikovat např. přes Ethernet pomocí hardwarových MAC adres. Pro adresu počítače v podsíti je zde vyhrazeno 15 bitů, tato síť tedy může mít až 2^15-1 = 32767stanic.

2.2.6 Router
Data, která jsou vyslána na síť jsou rozdělena do „balíčků“ velkých přibližně 1500 bytů. Tyto balíčky se nazývají packety. Každý packet obsahuje hlavičku ve které jsou mimo jiné IP adresa odesílatele, IP adresa příjemce a pozice v proudu packetů (aby bylo možné z packetů složit původní data). Router (směrovač) je zařízení které přesměrovává IP packety mezi jednotlivými uzly v síti. Každý packet může mezi dvěma počítači cestovat jinou trasou – podle aktuální propustnosti, funkčnosti routerů apod. Routery tvoří páteř Internetu – umožňují doručení packetů z jednoho počítače do kteréhokoliv počítače jinde. Konkrétně se může jednat o rozmanitý hardware od úrovně jednoduchého zařízení pro rozvedení Internetového připojení v domácnosti až po specializovaný hardware s výkonem superpočítače fungující na páteřních sítích. Může také jít o „obyčejné PC“ s více síťovými rozhraními a specializovaným software.

Obrázek 6: Příklady hardwarových routerů

Úkolem routeru tedy je přeposílat příchozí packety na takový svůj výstupní port který vede k cíli pro který jsou data určeny. Protože je rozsah Internetu obrovský nezná žádný router všechny ostatní routery v Internetu, zná pouze své více či méně omezené okolí. Pokud je packet určen pro síť kterou router zná odešle ji přímo do této sítě. Pokud cestu nezná, odešle ji na defaultní port, vedoucí obvykle k vyšší hierarchické jednotce v síti (například ISP – Internet Service Provider) nebo hraniční router autonomního systému ve kterém se router nachází. Vnitřně má tedy v sobě každý router konfigurační tabulku která mu ukazuje které jeho výstupní porty vedou ke kterým sítím, případně další informace jako priority těchto spojení apod. Tato tabulka může být sestavena buď staticky – ručně administrátorem serveru což je výhodné z hlediska rychlosti reakce routeru ale na druhou stranu pokud vypadne jediný prvek ve staticky zadané cestě stane se cílová síť této cesty pro packety zasílané na náš router nedostupná. Nebo může být sestavována (do jisté míry) automaticky – dynamicky. U těchto typů tabulek se každý router po připojení do sítě ohlásí svým nejbližším sousedům. Ti se mu ozvou nazpátek a ohlásí mu své sousedy. Router se pak ozve jim, ti mu opět ohlásí své sousedy atd. (pouze do jistého limitu v závislosti na použitém protokolu a jeho nastavení, např. protokol RIP podporuje max. 16 skoků). Poté je celý proces ohlašování se svým sousedům opakován v jistém časovém intervalu (např. 30 sekund). Nevýhodou je vyšší režie protože část přenosového pásma je zabírána těmito „servisními“ informacemi, výhodou je pružnost takové sítě – pokud vypadne jeden uzel je cesta k síti kam vedl automaticky nahrazena jinou funkční cestou (samozřejmě pokud je to možné a nebyl to jediný uzel propojující tuto síť s ostatními).

2.2.7 Autonomní systémy, BGP

Autonomní systém je kolekce IP systémů a routerů pod správou jedné entity, která navenek zachovává jednotnou směrovací politiku. Každý autonomní systém má přidělené unikátní identifikační číslo ASN (původně 16-bitové, tzn. max 65000 AS, od roku 2007 32-bitové). Toto číslo je využíváno protokolem BGP pro směrování IP provozu mezi autonomními systémy.

Border Gateway Protocol (BGP) je klíčovým protokolem pro určování dlouhých tras v Internetu. Pracuje na principu vytváření tabulek IP sítí nebo jejich prefixů ve kterých lze vyhledávat trasu mezi sítěmi na úrovni autonomních systémů (AS). Většina běžných uživatelů se s protokolem BGP nesetká přímo. Tento protokol využívají hlavně poskytovatelé internetového připojení – ISP (Internet Service Providers) aby si mezi sebou mohli směrovat

IP provoz. BGP využívá TCP port 179. Demonstrace jak pracuje BGP protokol je například na adrese http://bgplay.routeviews.org/bgplay/.
Potíže, se kterými se provoz BGP protokolu musí potýkat vyplývají z obrovského růstu Internetu. Až do roku 2001 totiž globální směrovací tabulka Internetu pro autonomní systémy rostla exponenciálně což by dříve či později vedlo k zhroucení konektivity. Fungování systému bylo tedy na základě dohody mezi ISP modifikováno tak aby došlo k agregaci co nejvíce cest v systému a tím se co nejvíce zpomalil růst globální směrovací tabulky. Na několik let bylo dosaženo lineárního růstu, který ale v roce 2004 opět přešel do růstu exponenciálního. 13. října 2006 dosáhla globální směrovací tabulka velikosti 200 000 položek (v současnosti – počátek roku 2008 asi 250 000 položek). K vylepšení agregace v globální GBP tabulce je využívána tzn. černá díra v síti. Například AS který má alokovaný adresní prostor 172.16.0.0/16 ze kterého ovšem využívá pouze bloky 172.16.0.0/18, 172.16.64.0/18 a 172.16.192.0/18. Takový AS pak často do globální BGP tabulky vystaví kompletní adresní prostor 172.16.0.0/16 čímž bude přijímat i packety určené pro „černou díru“ 172.16.128.0/18. Tyto packety ovšem v tichosti zahodí. Z tohoto stavu vyplývá mimo jiné i to že jako „páteřní“ routery Internetu lze použít hardware až od jisté technické úrovně výše. Mnoho routerů, obzvláště typu Small Office/Home Office (SOHO) v sobě podporu BGP protokolu ani neobsahuje. Některé vyspělejší ano, nicméně mohou mít například omezenou paměť pouze pro 20 000 položek což již pro globální BGP tabulku nestačí. Kromě BGP tabulky musí routery samozřejmě udržovat i tabulky pro své vlastní sítě, což u velkých ISP může představovat třeba i dalších 50% položek navíc.

2.2.8 Základní síťové nástroje v operačním systému Windows

ping
Slouží ke zjištění, zda je daný server „živý“ a jakou má latenci jeho spojení s námi

pathping
Vypočítá statistiku ping pro celou cestu

tracert
Vypíše trasu paketů mezi námi a cílovým systémem.

nslookup
Umožňuje zasílat dotazy přímo konkrétním DNS serverům, specifikovat jaké typy záznamů chceme vypsat a také kompletní přenos zóny.

netstat
Statistika síťového provozu – naslouchající porty, otevřená spojení apod. Parametr –n zobrazí adresy v IP podobě místo DNS názvů, parametr –a zobrazí všechna spojení a naslouchající porty, parametr –b spustitelný soubor spojený s daným spojením včetně PID procesu

arp
Pomocí přepínače „–a“ vypíše tabulku ARPzáznamů lokálního počítače, tzn. IP adresy stanic se kterými se náš systém pokusil spojit a příslušné MAC adresy. Parametrem „-s“ lze přidat ornamentní záznam.

net
Komplexní nástroj sloužící ke správě síťových prostředků (účty, počítače, soubory, skupiny, služby, sdílené prostředky). Tento příkaz se skládá ze dvou částí, slova net + druhého slova které může být: Accounts, Computer, Config, Continue, File, Group, Help, Helpmsg,

Localgroup, Name, Pause, Print, Send, Session, Share, Start, Statistics, Stop, Time, Use, User, View. Nápovědu k jednotlivým podpříkazům lze získat např. „net help accounts“. Jedná se o velmi důležitý příkaz, pomocí kterého lze provádět komplexní správu počítače. Příklady:

- Přidat z příkazové řádky uživatele: net user jmeno heslo /add
- Zobrazit počítače v místní pracovní skupině nebo doméně: net view
- Připojit síťovou složku jako místní logický disk: net use e: \\fileserver\Dokumenty
- Zastavit Windows službu: net stop jmeno_sluzby

Úkoly pro cvičení:

1. Zjištěte, jakou máte IP adresu a masku sítě. Kolik počítačů může být připojeno maximálně v síti, do které patříte?
2. Určete, zda máte veřejnou nebo privátní IP adresu.
3. Zjistěte na příkazové řádce název stanice (hostname)
4. Zjistěte, jaké máte otevřené otevřené porty a se kterými programy jsou spojeny.
5. Zjistěte, jaký máte nakonfigurovaný DNS server, otestujte jej, zda z něj lze provést přenos zóny (pomocí nslookup, volba „zobrazit všechny záznamy“).
6. Pomocí ipconfig zjistěte obsah své místní DNS cache. Pomocí stejného nástroje vyprázdněte DNS cache
7. Na Internetu najděte stránky provádějící reverzní DNS dotazy a proveďte jej na svou ip adresu (reverse DNS lookup)
8. Zjistit informace o doméně, do které patří, síť ve které se nacházíme (whois), číslo autonomního systému (např. robtex.com)
9. Zjistit trasu k nejznámějším serverům (google, seznam, ..)
10. Zjistit MAC adresu svého síťového adapteru. Najděte na Internetu databázi kódů výrobců (vendor MAC address lookup) a vložte do něj část MAC adresy označující výrobce. Zjistěte zda souhlasí s údaji v popisu síťového adaptéru
11. Zjistěte obsah ARP tabulky. Proveďte ping na některý z počítačů v místní síti (např. na IP adresy některého z jiných počítačů v učebně) a zjistěte obsah znovu. Proč se obsah tabulky změnil?
12. Pomocí příkazové řádky nasdílejte složku na místním disku
13. Pomocí příkazové řádky a příkazu telnet pošlete přes protokol smtp emailovou zprávu. Zfalšujte odesilatele zprávy. V přijatém emailu analyzujte hlavičky a ověřte, zda a do jaké míry je možné zkontrolovat identitu odesilatele. Jak by se situace změnila při použití digitálního podpisu?

Úkol pro seminární práci:

Zjistit a sepsat všechny výše uvedené informace o připojení doma (na koleji, v knihovně,.. tzn. jinde než ve škole)

3 Odposlouchávání sítě

Odposlouchávání síťového provozu znamená schopnost číst ze sítě pakety kterou nejsou určeny pro náš počítač. K tomu je potřeba jednak tzv. sniffer a jednak příslušný ovladač který přepne síťovou kartu do tzv. promiskuitního módu. Na operačním systému Linux, je to Pcap, jeho verze pro Windows se nazývá WinPcap. Nejznámějším snifferem je WinShark (dříve Ethereal), dostupný jak pro Linux tak pro Windows. Pro Windows dále existuje nástroj Microsoft Network Monitor přímo od Microsoftu nebo volně dostupný Packetyzer, který využívá stejné jádro jako Ethereal. „Klasickým" snifferem je také tcpdump.
Odposlech může sloužit k mnoha účelům – mezi ty legitimní patří analýza síťového provozu kvůli hledání poruch sítě, hledání aktivity červů nebo hackerů, vývoj software apod. Mezi nelegitimní patří například odchytávání hesel. V závislosti na konkrétním síťovém protokolu mohou být hesla v síťovém provozu buď v zašifrované nebo zakódované podobě nebo také ve formě čitelného textu. Ve volně čitelné podobě jsou zasílány hesla protokolů FTP, telnet, POP3, NNTP a dalších.
Síťový analyzátor se skládá z pěti základních části:
- **hardware**: většina síťových analyzátorů je ve formě software a pracují s běžnými operačními systémy a síťovými kartami (NIC – network interface card). Existují ale i specializované hardwarové síťové analyzátory které mají výhodu v tom že mohou detekovat i hardwarové chyby jako například CRC (cyclic redundancy check), problémy s napětím, kabely, jiter (časová základna), jabber (zařízení které nesprávně zpracovává elektrické signály), chyby při vyjednávání spojní. Některé síťové analyzátory podporují pouze Ethernet nebo wifi, jiné jsou schopné pracovat s více adaptéry a dovolí uživatelům měnit konfiguraci. Někdy je k provozu síťového analyzátoru potřeba také hub nebo cable tap pro připojení k existujícímu kabelu.
- **capture driver** (ovladač): tato část je konkrétně zodpovědná za zachycování síťového provozu na kabelu. Kromě toho filtruje pouze ten provoz který nás zajímá a ukládá ho do datového bufferu. Tvoří jádro síťového analyzátoru a bez něj je nemožné zachytávat data.
- **buffer** (zásobník): tato část ukládá zachycené data. Do bufferu se můžou ukládat buď tak dlouho dokud je v něm místo nebo rotačním systémem jako například „round robin" při kterém nejnovější data nahrazují nejstarší data. Buffery mohou být realizovány buď v paměti nebo na pevném disku.
- **real-timový analyzátor**: analyzuje příchozí data. Některé analyzátory to potřebují k tomu aby odhalili problémy na síti, IDS (intruder detection system) jej používají k odhalení signatur počítačového útoku
- **dekodér**: zobrazuje síťový provoz ve formě která je čitelná pro člověka. Dekodéry jsou specifické pro každý protokol. Různé síťové analyzátory se liší tím kolik protokolů jsou schopny dekódovat. Nové dekodéry jsou průběžně přidávány.

Legitimní použití sniferů:

- konverze dat na formát čitelný pro člověka
- řešení problémů na síti
- analýza výkonu sítě a hledání úzkých míst
- detekce průniků
- logování provozu pro forenzní analýzu a důkazy
- analýza provozu aplikací
- zjištění porouchané síťové karty
- zjišťování původu DoS (Denial of Service) útoku
- detekování spywaru
- debugování síťových aplikací ve fázi vývoje

- nalezení kompromitovaného systému
- ověřování dodržování firemní politiky pro využívání výpočetní techniky
- vzdělávací účely – praktické učení se fungování protokolů
- reverzní inženýrství protokolů za účelem napsání vlastního klienta a podpůrných programů

Nelegitimní použití:

- zachycování uživatelských jmen a hesel v čitelné formě
- kompromitování soukromých informací
- zachycování a znovupřehrávání Voice over IP telefonních konverzací
- mapování sítě
- pasivní OS fingerpriting

K tomu, aby mohl útočník odposlouchávat síť, musí napřed získat fyzický přístup ke komunikačním kabelům systému, který ho zajímá. To znamená, že se musí nacházet na stejném segmentu sítě nebo se napojit na kabel někde v cestě kudy informace prochází. Pokud narušitel není fyzicky přítomný ve stejném segmentu sítě jsou jeho možnosti pro její odposlouchávání silně omezeny, ale přesto nějaké existují:

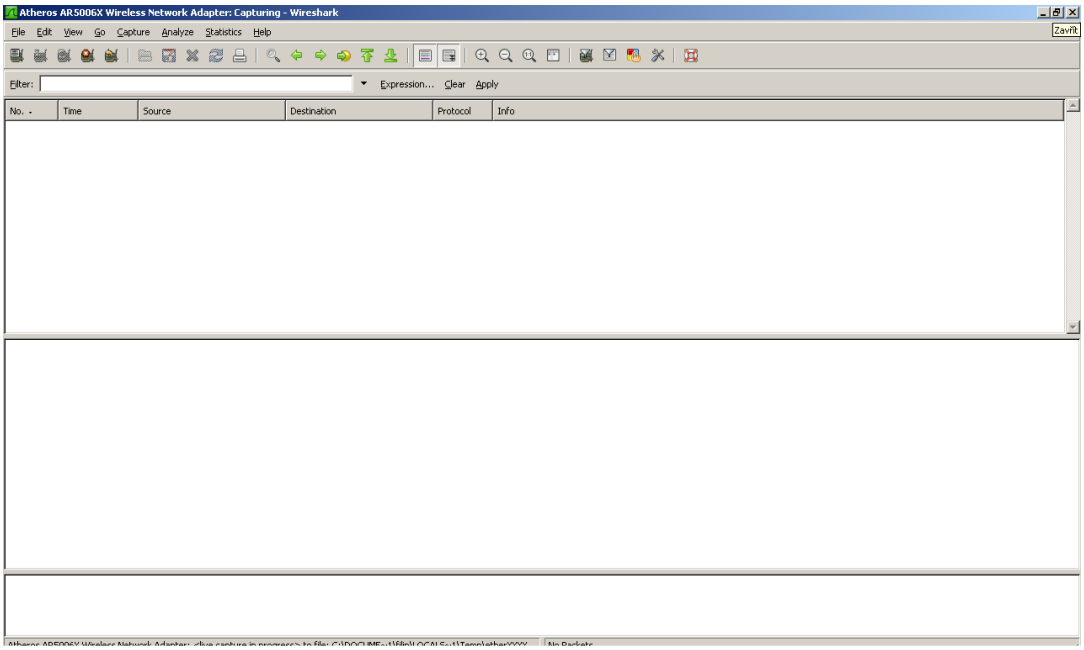
- nabourání se do cílového počítače a nainstalování dálkově ovládaného sniffery
- nabourání se do přístupového bodu sítě, jako například do počítačů ISP (Internet service provider) a instalace snifferu
- nalezení a získání přístupu do systému u ISP kde už sniffer je nainstalování
- využití technik sociálního inženýrství k získání fyzického přístupu k zařízením ISP a instalace snifferu
- získání insidera v cílové organizaci nebo v ISP který sniffer nainstaluje
- přesměrování trasy komunikace tak aby vedla přes počítač ovládaný narušitelem

Útočníci často používají sniffery k ovládání zadních vrátek (back door). Jednou z metod je instalace snifferu do systému ve kterém vyčkává a naslouchá v očekávání specifických informací. Příkazy pro backdoor jsou pak zaslány ovšem ne na systém ve kterém je backdoor a sniffer ale na systém který s nimi sousedí (je ve stejné síti). Protože sniffer odposlouchává provoz na celém segmentu sítě, příkazy samozřejmě zachytí a předá je backdooru. Tento typ zadních vrátek se velmi těžko odhaluje, protože z hlediska síťového provozu to vypadá, že napadený systém je síťový soused, který je v tom ovšem nevinně. Příklad takového backdoor snifferu je cd00r, který navíc pracuje v nepromiskuitním režimu čímž je ještě těžší jej odhalit. Za použití nástroje jako je například Nmap stačí odeslat sérii TCP SYN paketů na několik dohodnutých portů a backdoor potom otevře předkonfingurovaný port. Více informací o cd00r lze nalézt na www.phenoelit.de/stuff/cd00r.c.

3.1 Nejznámější síťové analyzátory

3.1.1 Wireshark (Ethereal)

Jeden z nejlepších snifferů. Je vyvíjen jako open source, přitom ale nabízí kvalitu komerčních produktů. Má velký počet funkcí, grafické rozhraní, dekodéry pro více než 1000 protokolů a je průběžně vyvíjen a spravován. Je dostupný jak pro UNIXové tak pro Windows systémy. Program je nyní přejmenovaná na WireShark. Domovská stránka je www.wireshark.org (www.ethereal.com).



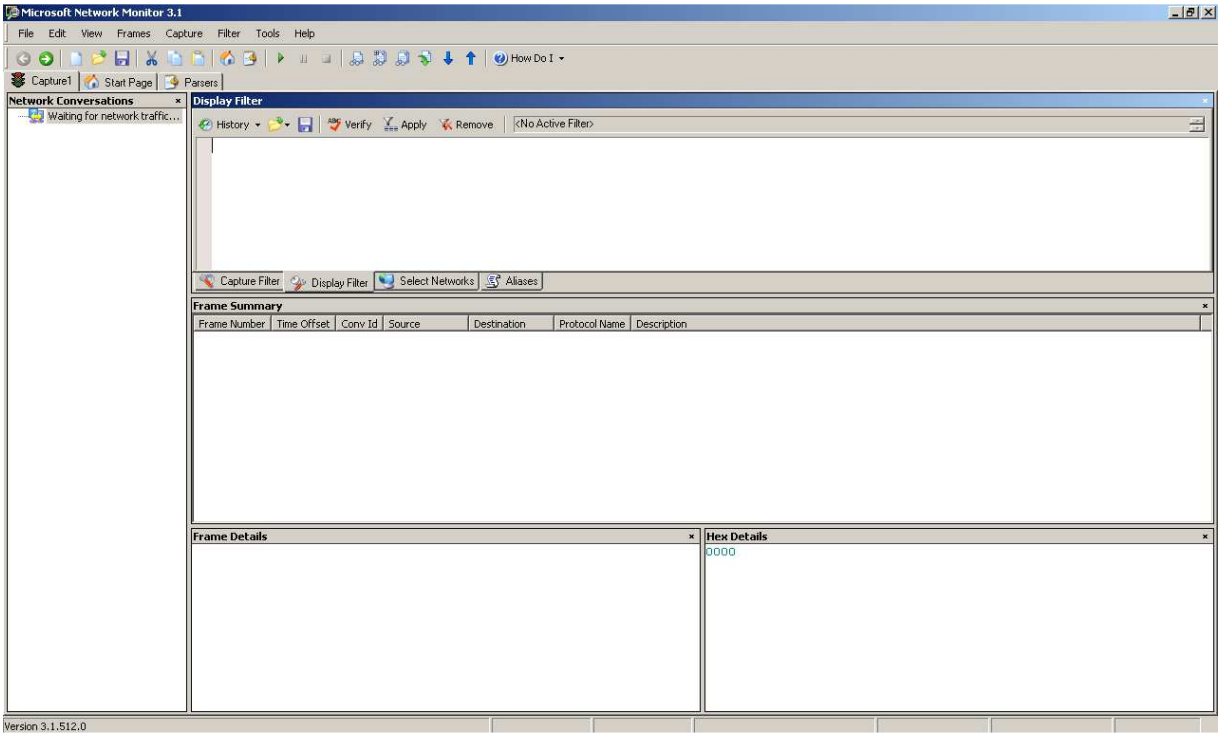
Obrázek 7: Wireshark

3.1.2 WinDump

Jedná se o Windows verzi populárního UNIXovského tcpdump. Využívá WinPcap knihovnu. Network Associates Sniffer. Jeden z nejpopulárnějších komerčních produktů. Nyní je prodáván společností McAfee Network Protection Solutions.

3.1.3 Microsoft Network Monitor

Windows 2000, Windows 2003, Windows 2008 mají jako volitelnou součást vestavěný sniffer od Microsoft. Je dostupný ve složce Administrativní nástroje ale je nutné ho doinstalovat z cd. Je také volně ke stažení na www.microsoft.com.



Obrázek 8: Microsoft Network Monitor

3.1.4 EtherPeek
Komerční sniffer od firmy WildPackets. Existují verze pro Windows i pro Mac. www.wildpackets.com.

3.1.5 Tcpdump
Nejstarší a nejrozšířenější síťový sniffer. Pracuje na příkazovém řádku v UNIXových systémech, vyvíjí jej Network Research Group (NRG) z Information and Computing Scieneces Division (ICSD) v Lawrence Berkeley National Laboratory (LBNL). www.tcpdump.org.

3.1.6 Snoop
Sniffer ovládány pomocí příkazové řádky, je součástí operačních systémů Sun Solaris. Vyniká v práci s protokoly specifickými pro tuto platformu.

3.1.7 Sniffit
Snifer ovládaný z příkazové řádky, běží na OS Linux, SunOS, Solaris, FreeBSD a IRIX.

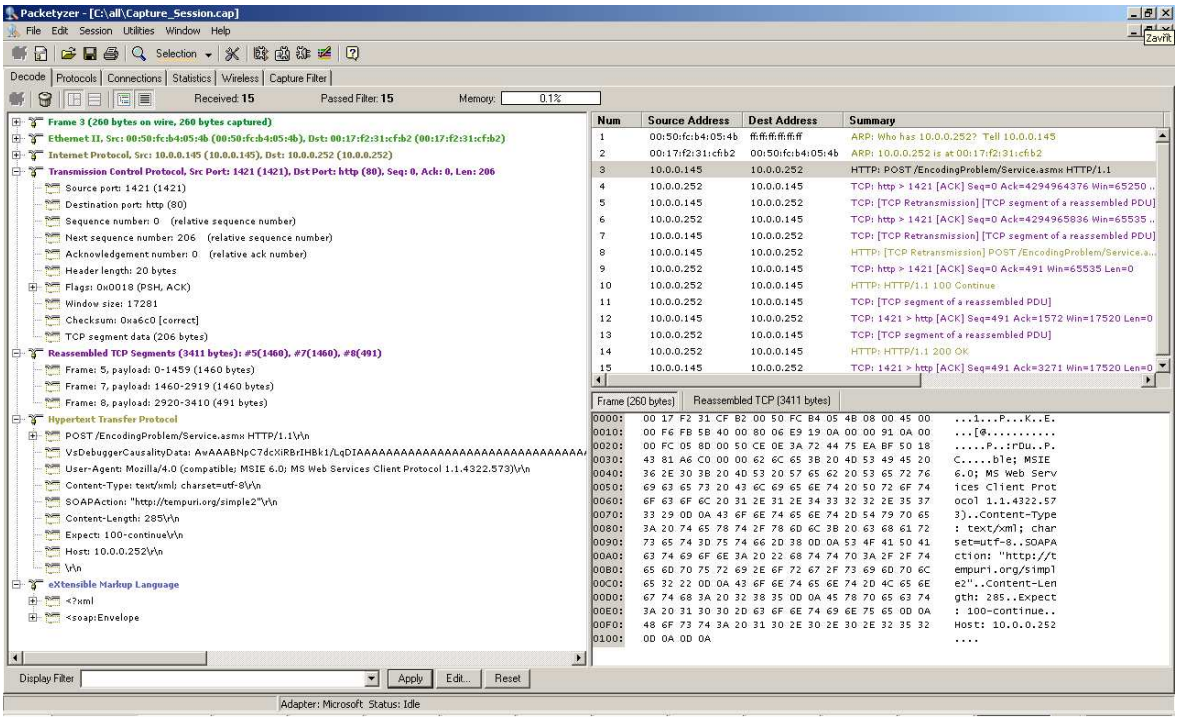
3.1.8 Snort
Spíše než o sniffer se jedná o Network Intrusion Detection System (IDS) – systém pro detekci průniků do sítě, jeho klíčovou součástí je sniffer. Je aktivně vyvíjený a dostupný na www.snort.org jak pro Linux tak pro Windows.

3.1.9 Dsniff
Velmi populární sniffovací balíček – kolekce programů které jsou určeny na vyhledávání informací jako jsou hesla a na odposlouchávání na přepínané síti. Pro Linux i pro Windows.

3.1.10 Ettercap
Sniffer speciálně navržený pro odposlouchávání na přepínané síti. Má vestavěné funkce jako sběr hesel, OS fingerprinting (pasivní identifikace operačních systémů), modifikace síťového provozu (character injection). Běží na několika platformách včetně Linuxu, Windows, Solária. Dostupný na http://ettercap.sourceforge.net.

3.1.11 Analyzer
Volně dostupný sniffer pro OS Windows který je aktivně vyvíjen tvůrci WinPcap a WinDump. Ke stažení na http://analyzer.polito.it.

3.1.12 Packetyzer
Volně dostupný sniffer pro OS Windows, vnitřně používá jádro Etherealu. Většinou bývá o jednu nebo dvě verze pozadu proti Etherealu. Nabízí o něco lepší grafické rozhraní. Je dostupný na www.networkchemistry.com.



Obrázek 9: Packetyzer

3.1.13 Linsniff

Specializovaný sniffer pro linux, který slouží pouze k vyhledávání a zachycování hesel všemožných protokolů.

3.1.14 Carnivore

Carnivore (masožravec) je software který není veřejně dostupný, ale o kterém se v bezpečnostní komunitě hodně diskutuje. Jedná se totiž o původní kódové označení snifferu který byl vytvořen speciálně pro americkou FBI. Slouží pro odposlouchávání komunikace mezi vybranými jedinci při odhalování zločinu. Jméno snifferu bylo později změněno na DSC100 ve snaze jej dostat z očí veřejnosti. Použití tohoto snifferu vypadá většinou tak že agenti FBI přijedou se soudním příkazem do sídla ISP, kde do sítě zapojí svojí „černou skříňku“ na které běží obvykle Windows 2000 s přeinstalovaným Carnivore. Carnivore potom zachycuje hlavičky veškeré komunikace od/pro podezřelého. Četnost nasazení Carnivore vzrostla po 11. září 2001.

FBI si nechala Carnivore vytvořit na zakázku protože jiné ať už volně dostupné nebo komerční nástroje nevyhovovaly potřebám policejních složek. Například každá větší emailová zpráva je při přenosu po síti rozdělena na více paketů a v cíli opět složena. Utility jako mailsnarf ale zprávu samy složí do původní formy, což je u kriminálního vyšetřování problém neboť u soudu lze pak napadnout že se jedná už o vykonstruovanou informaci neboť není zaručeno že je složená zpráva autentická kvůli možnému výpadku nebo „přebývání“ paketů po cestě. Carnivore tedy zachycuje pouze přímé pakety včetně původních čísel sekvencí, portů, časových údajů a neskládá je dohromady čímž je zaručena věrohodnost celého systému protože jakékoliv chybějící nebo nadbývající data lze okamžitě odhalit.

Další problém který by FBI měla s běžnými sniffery je úmyslná minimalizace odposlouchávaných dat. Když je odposlouchávána například telefonní linka, musí být u monitoru přítomen člověk který komunikaci kontroluje a pokud na ní probíhá hovor (například manželka sledovaného) který nesouvisí s podezřelým vypne nahrávání. Na podobném principu musí pracovat Carnivore – nezachycuje nic kromě toho¨, co souvisí se sledovanou osobou.

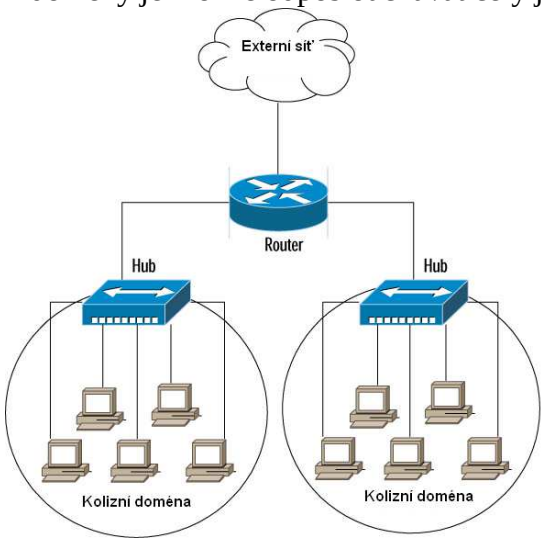
3.2 Hardware

3.2.1 Cable tap

Tap znamená Test Access Point, jedná se zařízení které umožní číst data z jakéhokoliv síťového kabelu mezi počítači, huby, switchi, routery. V podstatě nám umožňují „napíchnout“ kabel. Nejznámější výrobci jsou Net Optics (www.netoptics.com) a Century Tap (www.shomiti.net/shomiti/century-tap.html). K dispozici jsou tapy jak pro měděné tak pro optické kabely.

3.2.2 Hub

Hub je rozvodný síťový prvek, který umožňuje zapojit více zařízení na jedno sdílené médium. Využívá se například v sítích Ethernet. Když jeden počítače v síti zašle informaci na hub, hub ji „slepě“ rozešle všem ostatním počítačům, které jsou k němu připojeny. Každý z počítačů si pak podle cílové MAC adresy v daném datagramu a podle své vlastní MAC adresy určí jestli paket přečte nebo zahodí. Oblast sítě, ve které hub takto přeposílá data se nazývá kolizní doména nebo broadcast (vysílací) doména. Velké kolizní domény zpomalují provoz na síti. V kterémkoliv bodě kolizní domény je možné odposlouchávat celý její provoz.

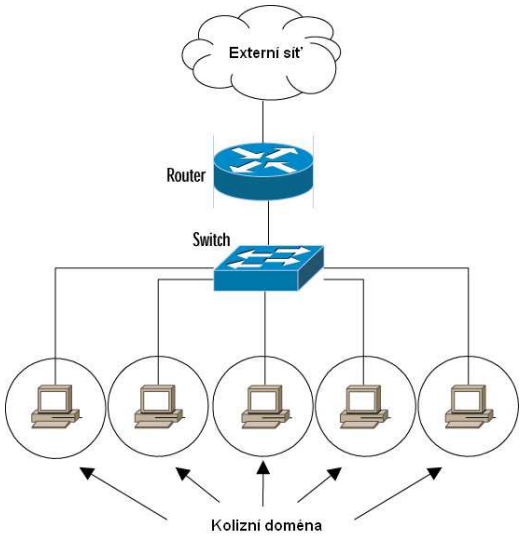


Obrázek 10: Kolizní doména routeru

3.2.3 Switch

Switch je také centrální síťový prvek na stejné OSI vrstvě jako hub ale pracuje jiným způsobem. Když obdrží data od počítače nepřepošle je všem ostatním počítačům v síti ale pouze tomu který má MAC adresu uvedenou jako cílovou adresu datagramu. Vnitřně si tedy switch musí udržovat tabulku ve které má ke každému svému portu přiřazenou MAC adresu počítače připojeného na tento port. Tím je zúžena kolizní doména na jediný počítač, ušetřeno přenosové pásmo a znemožněno odposlouchávání síťového provozu ostatních počítačů ve stejné síti prostým přepnutím síťové karty do promiskuitního módu. Díky klesajícím cenám a pokroku v technologiích se huby v nově budovaných sítích již nepoužívají, switche je plně nahradily.

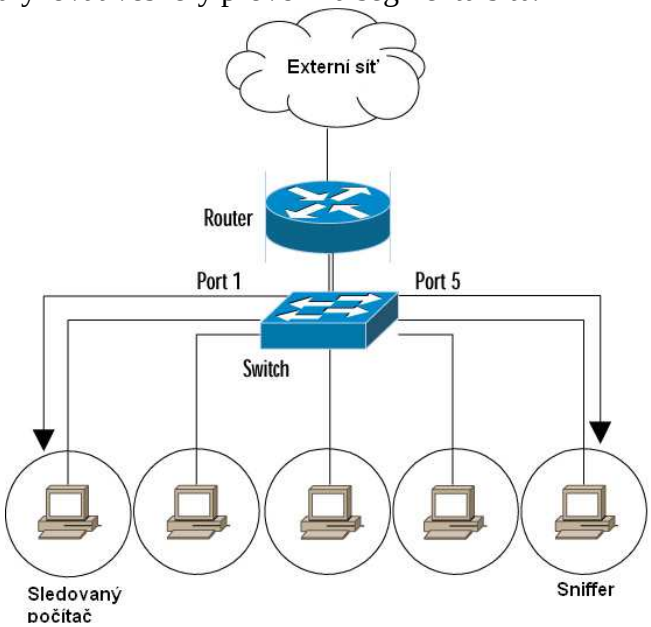
I v přepínané síti existují určité typy síťového provozu, které lze odposlechávat na všech stanicích – broadcast vysílání a ARP protokol.



Obrázek 11: Kolizní doména switche

3.2.4 Legitimní analýza síťového provozu na přepínané síti

I na přepínané síti je samozřejmě potřeba provádět analýzu síťového provozu z legitimních důvodů – správa sítě, řešení problémů, vývoj programů apod. K tomu slouží tzv. port mirroring (u technologií firmy Cisto nazývaný port spanning) – zrcadlení portů. Většina switchů a routerů tuto technologii podporuje. Při zrcadlení portů je switch rekonfigurován tak aby duplikoval provoz z portu který chceme monitorovat na port na kterém běží sniffer. Přesné nastavení port mirroringu se liší případ od případu, je popsáno v dokumentaci každého konkrétního switche. Pokud administrátor switche zrcadlí uplink port switche (příchozí spojení) může pak analyzovat veškerý provoz na segmentu sítě.



Obrázek 12: Odposlech sítě

3.2.5 Techniky nežádoucího odposlouchávání provozu na přepínané síti

Odposlouchávání na přepínané síti by správně nemělo být možné. V praxi to někdy možné je, i když ne vždy. Záleží na schopnostech switche, jeho konfiguraci, chybách v jeho firmware apod. Existují tyto útoky:

3.2.6 MAC flooding

Na switch jsou zaslány miliony falešných MAC adres. Některé switche je tímto možné zahltit natolik že potom přepnou do tzv. fail-over módu ve kterém se switch začne chovat stejně jako hub – přeposílat provoz z každého portu do všech ostatních portů. Tento útok je poměrně brutální, lehce zjistitelný. Populární nástroj k jeho provedení je Macof, součást balíčku Dsniff pro linux (http://monkey.org/~dugsong/dsniff).

3.2.7 MAC duplicating (MAC cloning)

Při tomto druhu útoku si nastavíme MAC adresu vlastního počítače tak aby byla stejná jako MAC adresa počítače který chceme odposlouchávat. Některé starší nebo méně sofistikované switche potom začnou posílat síťový provoz na obě rozhraní zároveň. Použít se dá například nástroj SMAC (pro Windows), případně některé síťové adaptéry umožňují změnit MAC adresu přímo. Jak ale zabránit tomu aby oběť vysílala vlastní ARP hlášení a tím došlo k opravení ARP tabulky na switchi? Jedna možnost je využít některého DoS (Denial of Service) útoku k tomu aby byla oběť vyřazena z provozu (takto bude ale možné se pouze vydávat za oběť, ne ji odposlouchávat). Pak se lze ale pouze vydávat za oběť, nikoliv ji odposlouchávat.

3.2.8 ARP redirect

Pokud počítač potřebuje zjistit MAC adresu jiného počítače v místní síti, vyšle takzvaný ARP požadavek. Každý počítač si udržuje ARP tabulku (na OS Windows lze vypsat příkazem „arp –a“ v příkazovém řádku) ve které jsou uchovávány MAC adresy ostatních počítačů se kterým měl navázané spojení. ARP zprávy jsou přes switch vysílány všesměrově takže všechny počítače v daném segmentu vidí jak požadavek tak odpověď. Útočník vysílá všesměrové ARP zprávy ve kterých se vydává za bránu. Tím pádem je veškerý provoz směřován přes něj. Nebo může poslat ARP odpověď ve které se vydává za bránu poslat pouze jednomu počítači, pak bude přes útočníka procházet provoz pouze tohoto počítače (útočník musí v obou případech data dále přeposílat na pravý router aby nedošlo k jeho odhalení kvůli tomu že oběti přestane fungovat spojení ven ze sítě).

3.2.9 ICMP redirect

V některých případech jsou počítače na stejném fyzickém segmentu sítě – na stejném switchi ale jsou v rozdílných logických segmentech – tzn. jsou v různých IP podsítích. Když se počítač A chce spojit s takto zapojeným počítačem B, zašle požadavek na router. Router ví že jsou oba na stejném fyzickém segmentu, takže zašle ICMP Redirect příkaz na počítač A ve kterém mu dává na vědomí že pakety pro počítač B může zasílat přímo pomocí MAC adres. Útočník – počítač X může zaslat falešný ICMP redirect do počítače A, ve kterém tvrdí že pakety pro počítač B může posílat na MAC adresu X (X se tedy vydává za B).

3.2.10 ICMP Router Advertisements

ICMP Router Advertisements informují počítače kdo je v síti router. Útočník může rozesílat tyto zprávy a tvrdit že je router a počítače začnou forwardovat všechny IP pakety přes něj.

3.2.11 MAC Spoofing

Útočník předstírá že je jiný počítač tím že si přivlastní jeho MAC adresu. Udělá to tak že zašle falešné ARP odpovědi na Switch, který si ve své MAC tabulce nabudí napadený počítač za útočníka. Provoz určený pro oběť pak bude chodit útočníkovi. Ten ji pak všesměrové přeposílá dál aby se dostala k oběti. Nelze se takto vydávat za oběť, ale lze ji takto odposlouchávat. Některé switche mají protiopatření, které umožňuje přiřadit MAC adresy na

jednotlivé porty staticky (ručně). To zamezí MAC spoofingu, nicméně je to použitelné pouze u menších sítí, u velkých by to bylo příliš pracné.

3.2.12 Rekonfigurace port mirroringu na switchi

Jak bylo zmíněno dříve port mirroring (nazývaný také port spanning) lze legitimně využít k monitorování provozu ostatních. Útočník se může pokusit připojit na switch například Telnetem (většinou je potřeba znát heslo) a překonfigurovat ho.

3.2.13 Fyzický přístup

Pokud má útočník fyzický přístup ke switchi nebo ke kabelům může použít cable taps (hardwarový odposlech kabeláže).
Metod je tedy více, ovšem ne každá bude fungovat, zvláště na novějších typech switchů – výrobci jejich slabiny obvykle znají snaží se je opravovat. Dobrý zdrojem informací o monitorování provozu v přepínané síti je například FAQ soubor k programu Dsniff (http://monkey.org/~dugsong/dsniff/faq.html. Oblíbenými nástroji pro provedení těchto útoků jsou již zmíněný Dsniff a CAIN.

3.2.14 Detekce snifferů

Odposlouchávání provozu na síti je pasivní typ útoku. Provoz není nijak měněn, nejsou vysílány žádné informace. Přesto je možné odposlouchávání detekovat. Nejjednodušší metodou je kontrola zda je síťová karta nastavena v promiskuitním módu. Na linuxu lze použít příkaz „ifconfig –a“, případně příkaz „ip link“.
Detekování promiskuitního módu na OS Windows je těžší neboť zde nejsou žádné standardní příkazy který by vypsaly tento typ informací. Existuje ale zdarma dostupný nástroj PromiscDetect (http://ntsecurity.nu/toolbox/promiscdetect).
Některé sofistikovanější sniffery nicméně dokáží zakrýt své stopy tím, že zamaskují příznaky promiskuitního módu v operačním systému. Potom je nutné použít některou z následujících metod:

- monitorovat reverzní DNS lookup
- zaslat TCP/IP pakety na všechny IP adresy v Ethernetovém segmentu ale s falešnými MAC adresami.
- využít princip fungování ARP
- pečlivě monitorovat porty hubů.
- použít honeypot (nástrahu).

Nástroje pro detekci snifferů

- PromiScan
- AntiSniff
- Sentinel
- Neped
- Check Promiscuos Mode
- Ifstatus
- Promisc.c

3.2.15 Ochrana proti odposlechu komunikace

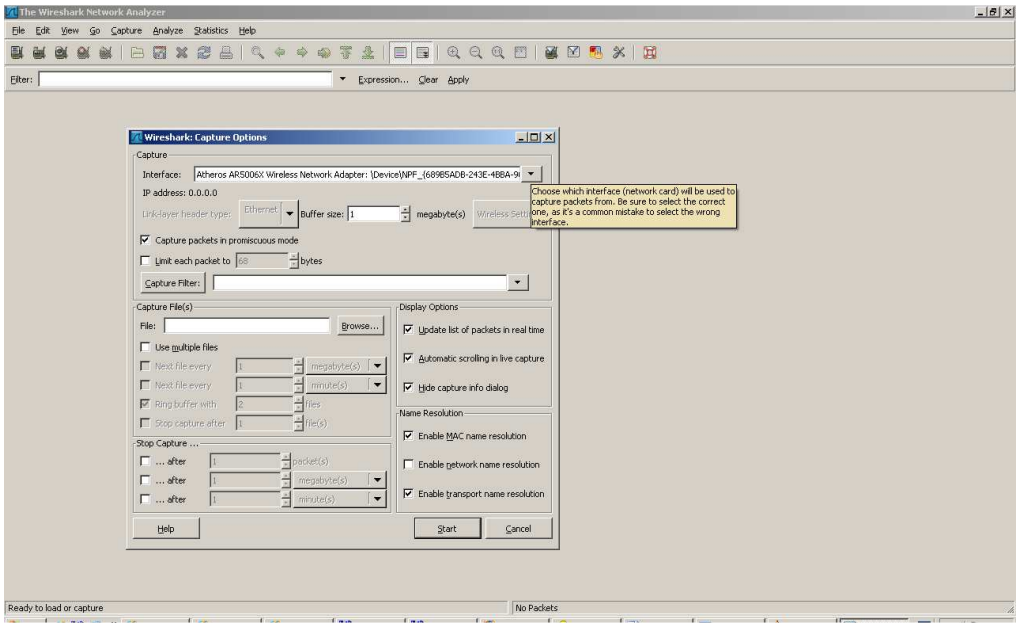
V podstatě jedinou ochranou proti odposlechu je šifrování komunikace, spolehnout se na zabezpečení přenosového kanálu jako takového nelze. Používá se celá řada protokolů, které využívají principy asymetrické kryptografie (veřejný a soukromý klíč, certifikáty, digitální

podpisy). Asymetrická kryptografie byla navržena právě proto, aby bylo možné bezpečně přenášet data po nezabezpečeném informačním kanále.

- SSH – Secure Shell. Funguje na aplikační vrstvě. Často je používán ke vzdálené správě serverů přes zabezpečenou příkazovou řádku. Hlavičky SSH nejsou zakódované, takže útočník může vidět aspoň zdrojovou a cílovou adresu.
- SSL (TSL) – Secure Socket Layer/Transport Layer Security. Poskytuje šifrování na transportní vrstvě, zabaluje protokoly vyšších vrstev (http, FTP, NNTP, POP3, IMAP)
- IPsec – IP security. Pracuje na síťové úrovni, používá rozšíření standardních IPv4 a IPv6 protokolů. Díky tomu je možné šifrovat každý protokol nad touto úrovní. Může pracovat v tunelovém módu, který poskytuje záměny IP hlaviček a tím maskuje originální zdrojovou a cílovou IP adresu.
- PGP – Pretty Good Privacy. Používá se hlavně pro šifrování a elektronický podpis mailové komunikace.
- OTP – One Time Passwords, jednorázová hesla. Obecná technika kdy jsou v aplikacích využívány hesla na jedno použití. Tím ztrácí odposlech hesel smysl (kromě útoku typu man in the middle).

3.2.16 Použití snifferu WireShark (Ethereal)

Současná verze WireSharku (0.99.7) podporuje 911 protokolů.



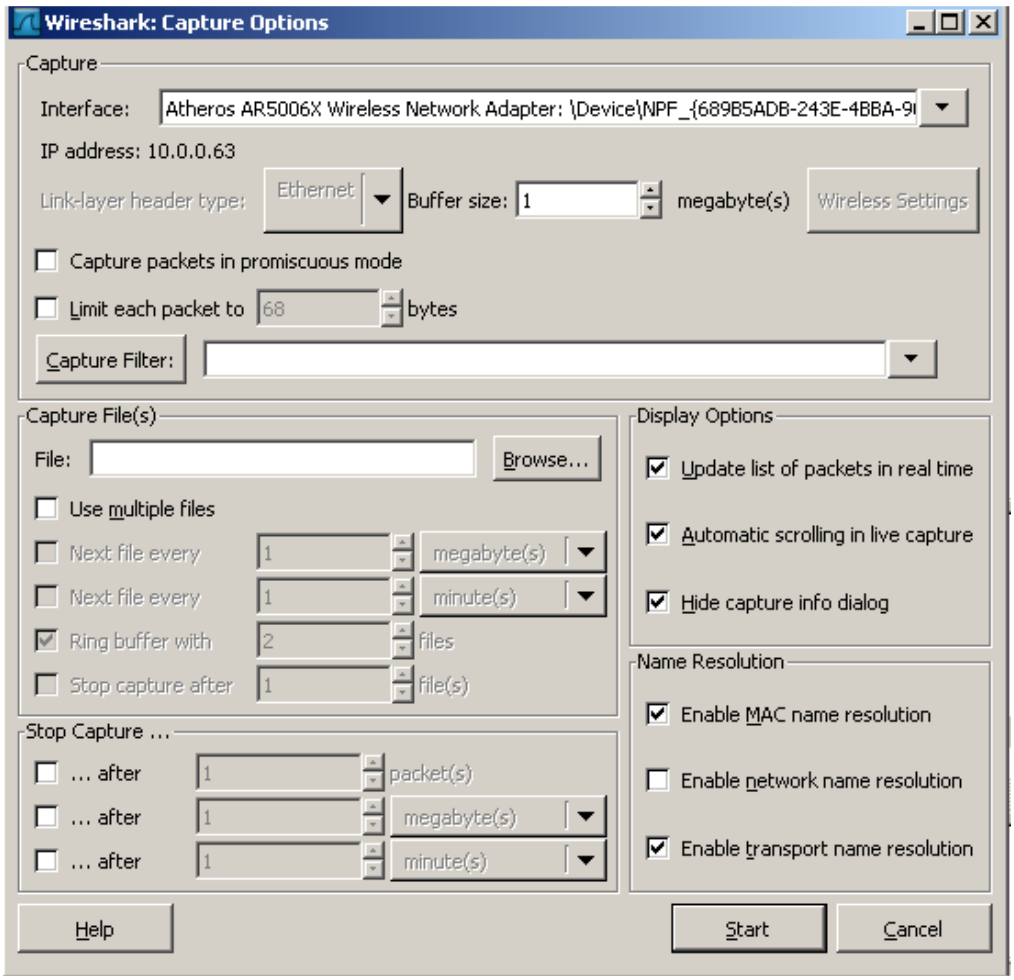
Obrázek 13: Výběr síťového rozhraní

Loopback na Windows

V operačním systému Windows nelze odposlouchávat adresu 127.0.0.1 (tzn. tzv. loopback provoz, například provoz mezi serverovou a klientskou aplikací běžící na tom samém počítači) protože Windows nemají loopback rozhraní. Toto virtuální rozhraní lze sice doinstalovat (návod pro instalaci virtuálního loopback na Windows XP zde: http://support.microsoft.com/kb/839013) ale jeho chování je jiné než chování klasických loopback rozhraní například v unixových systémech. Toto rozhraní nemá adresu 127.0.0.1 ale svou vlastní IP. Pokud adresu tohoto rozhraní pošleme ping a pokusíme se tento provoz zachytit, nepovede se to, Wireshark uvidí pouze arp dotazy.

Filtry
Přímé zachytávání síťového provozu poskytne obrovské množství paketů. Z těchto surových dat je nutné dostat pouze to co je pro nás podstatné. Ethereal umožňuje filtrovat jednak to které pakety jsou zachyceny (capture filter) a jednak to které ze zachycených paketů jsou zobrazeny (display filter).

Capture filtry
Filtrovací pravidla pro zachytávání paketů používají knihovnu pcap, stejně jako program tcpdump, takže s ním mají společnou syntaxi. Tcpdump filtr umožňuje třídit data mnoha způsoby. Jeho jazyk je navržen tak aby umožňoval pracovat hlavně s nejvíce používanými položkami z 2. vrstvy (síťová) a TCP/IP protokoly. Také je možné vyhledávat libovolné bitové sekvence v paketech. Capture filtr se nastavuje v dialogu dostupném přes Capture/Options nebo klávesovou zkratku Ctrl+K.



Obrázek 14: nastavení zachytávacího filtru

Zachytávací filtr se píše do pole „Capture filter“ jako textový řetězec sestavený podle následujících možností:

Adresa nebo název zařízení
Zřejmě nejpoužívanější filtr, vybírá pouze pakety z/pro určitý počítač a to podle IPv4 adresy, IPv6 adresy nebo DNS názvu:

```
host 192.168.1.1
host 2::8100:2:30a:c392:fc5a
```

```
host wizard
host www.ethereal.com
```

V případech kdy DNS překlad jména počítače dodá více než jednu IP adresu, jsou použity všechny vrácené adresy. Pokud chceme omezit výběr pouze na zdrojovou nebo cílovou adresu, přidáme modifikátory src (source - zdroj) a dst (destination – cíl):

```
src host 192.168.1.1
dst host 192.168.255.255
```

Hardwarové (MAC) adresy

Použijeme modifikátor ether:

```
ether host ff:ff:ff:ff:ff:ff
```

Modifikátory src a dst jdou použít také.

Porty

Klíčové slovo port omezí zachytávání pouze na pakety z/do určitého portu. Dále jsou k dispozici modifikátory tcp, udp a také src a dst.

```
port 80
tcp port 80
udp src port 53
```

Logické operátory

Jazyk tcpdump filtru umožňuje zadávat více podmínek filtrování najednou a strukturovat je pomocí logických operátorů: **not, and, or**. Tam kde je potřeba přetížit prioritu operátorů použijeme závorky. Pokud nepoužijeme závorky, jsou podmínky vyhodnocovány prostě zleva doprava.

Příklady:

```
not port 53
```

(vše kromě DNS přenosů)

```
host www.ethereal.com and ( port telnet or port ssh )
```

(hostitel www.ethreal.com a z něj pouze porty telnetu nebo ssh)

```
host www.ethereal.com and port telnet or port ssh
```

(hostitel www.ethereal.com a zároveň port telnetu nebo port ssh – ten ale odkudkoliv)

Protokoly

Filtr protokolů dovolí zachytit pouze ten protokol který potřebujeme. Jednotlivé protokoly jsou přístupné přes klíčové slova. Jsou to (není zde http!):

- **aarp** AppleTalk Address Resolution Protocol
- **ah** Authentication Header
- **arp** Address Resolution Protocol
- **atalk** AppleTalk
- **clnp** Connectionless Network Protocol
- **decnet** Digital Equipment Corporation Network protocol suite
- **esis** (or “es-is”) End System-to-Intermediate System
- **esp** Encapsulating Security Payload
- **icmp** Internet Control Message Protocol
- **icmp6** Internet Control Message Protocol, for IPv6
- **igmp** Internet Group Management Protocol
- **igrp** Interior Gateway Routing Protocol
- **ip** Internet Protocol
- **ip6** Internet Protocol version 6
- **ipx** Internetwork Packet Exchange

- **isis** (or “is-is”) Intermediate System-to-Intermediate System
- **iso** International Organization for Standardization
- **lat** Local Area Transport
- **mopdl** Maintenance Operation Protocol
- **moprc** Maintenance Operation Protocol
- **netbeui** NetBIOS Extended User Interface
- **pim** Protocol Independent Multicast
- **rarp** Reverse Address Resolution Protocol
- **sca** Systems Communication Architecture
- **sctp** Stream Control Transmission Protocol
- **stp** Spanning Tree Protocol
- **tcp** Transmission Control Protocol
- **udp** User Datagram Protocol
- **vrrp** Virtual Router Redundancy Protocol

Příklady:

```
imcp
```

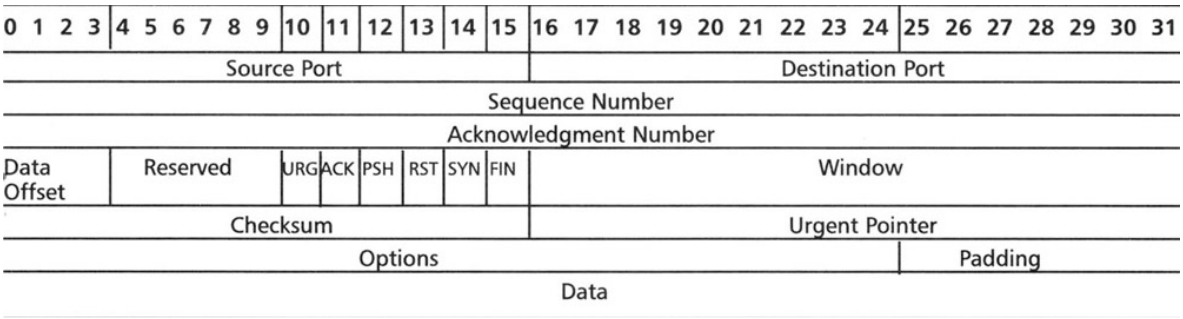
(pouze pakety protokolu ICMP)

```
not ipx
```

(vše kromě IPX paketů)

Pole protokolů

Zatímco protokoly samotné jsou přístupné přes klíčové slova, jednotlivé pole každého z protokolů takto jednoduše testovat nejdou. Pouze několik málo polí z hlaviček nejpoužívanějších protokolů má své klíčové slovo. Místo toho musíme využít jinou schopnost tcpdump jazyka - pokud je poloha hledaného pole známá, lze toho využít přes přímý přístup do bytů paketu. K získání konkrétního bytu z paketu napíšeme za název protokolu hranaté závorky a do nich offset (posunutí) bytu který nás zajímá od počátku paketu. Offsety začínají na nule, takže filtr tcp[0] vrátí první byte z tcp hlavičky, tcp[1] druhý byte atd.



Obrázek 15: Hlavička TCP protokolu (jednotky nahoře jsou v bitech)

Kromě jednobytového celého čísla můžeme vytáhnout také dvoubytové nebo čtyřbytové celé číslo, tím že do hranatých závorek dáme dvojtečku a za ni označení velikosti integeru: tcp[0:2], tcp[0:4] apod. Vícebitové hodnoty jsou uloženy v pořadí které se používá na síti, tedy big-endian. K výpočtu hodnoty vícebytového integeru se používají následující vzorečky:

- **2-byte** hodnota = byte0 * 0x100 + byte1
- **4-byte** hodnota = byte0 * 0x1000000 + byte1 * 0x10000 + byte2 * 0x100 + byte3

Čísla s předponou 0x jsou zapsána v hexadecimálním tvaru. Extrahovat byty lze bohužel pouze z některých protokolů. Jsou to tyto:

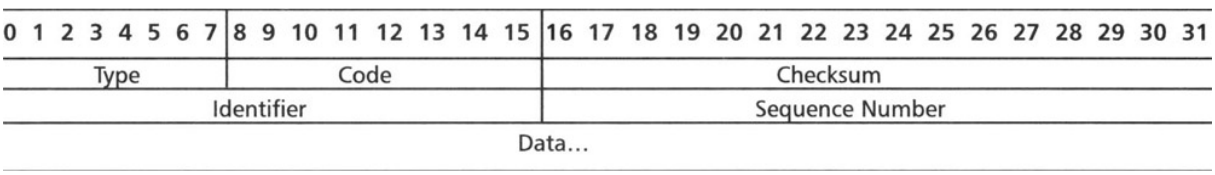
- **arp** Address Resolution Protocol
- **atalk** Appletalk
- **decnet** Digital Equipment Corporation Network protocol suite
- **ether** Ethernet
- **fddi** Fiber Distributed Data Interface
- **icmp** Internet Control Message Protocol
- **igmp** Internet Group Management Protocol
- **igrp** Interior Gateway Routing Protocol
- **ip** Internet Protocol
- **lat** Local Area Transport
- **link** Link layer
- **mopdl** Maintenance Operation Protocol
- **moprc** Maintenance Operation Protocol
- **pim** Protocol Independent Multicast
- **ppp** Point-to-Point Protocol
- **rarp** Reverse Address Resolution Protocol
- **sca** Systems Communication Architecture
- **sctp** Stream Control Transmission Protocol
- **tcp** Transmission Control Protocol
- **tr** Token-Ring
- **udp** User Datagram Protocol
- **vrrp** Virtual Router Redundancy Protocol

Hodnoty, které získáme z hlaviček paketů jsou typu integer a proto s nimi můžeme provádět následující operace porovnání:

```
>     větší než
>=    větší nebo rovno
<     menší než
<=    menší nebo rovno
=, == rovno
!=    není rovno
```

Dále jsou k dispozici aritmetické operátory +,-,* a / a bitové operátory & a |.

Příklad:
Klíčové slovo icmp způsobí zachytávání pouze icmp paketů. Těch je ale několik typů. To o který typ se jedná je určeno v poli „type“ hlavičky icmp. Hodnota 8 znamená typ icmp echo request (ping), hodnota 0 znamená icmp echo reply (odpověď na ping).



Obrázek 16: Hlavička ICMP paketu

Pokud tedy budeme chtít zachytávat pouze pakety icmp pingu, pak na to použijeme následující filtr:

```
icmp[0] == 8 or icmp[0] == 0
```

V případě ICMP protokolu máme navíc k dispozici i symbolické označení konstant. Použijeme je následujícím způsobem:

```
icmp[icmptype] == icmp-echo or icmp[icmptype] == icmp-echoreply
```

Následující tabulka uvádí přehled typů ICMP paketů:

icmp-echoreply	0
icmp-unreach	3
icmp-sourcequench	4
icmp-redirect	5
icmp-echo	8
icmp-routeradvert	9
icmp-routersolicit	10
icmp-timxceed	11
icmp-paramprob	12
icmp-tstamp	13
icmp-tstampreply	14
icmp-ireq	15
icmp-ireqreply	16
icmp-maskreq	17
icmp-maskreply	18

Tabulka 1: typy ICMP paketů

Bitové operátory

Bitové operátory se nám hodí tam, kde jsou pole v hlavičce nějakého protokolu uloženy jako bitové pole, například TCP protokol má 8-bitové pole nazvané flag (příznaky), ve kterém jsou individuální bity použity jako jednotlivé příznaky které jsou buď zapnuté (1) nebo vypnuté (0).

Keyword	**Value**
tcp-fin	0x01
tcp-syn	0x02
tcp-rst	0x04
tcp-push	0x08
tcp-ack	0x10
tcp-urg	0x20

Tabulka 2: konstanty představující TCP příznaky

Tyto příznaky jsou velmi užitečné při hledání problémů s firewallem nebo NAT, proto jsou i pro ně definovány symbolické konstanty. Například pokud chceme zachytávat pouze TCP SYN paket, použijeme následující:

```
tcp[tcpflag] == 0x02
```

což je ekvivalentní

```
tcp[tcpflag] == tcp-syn
```

Pokud bychom chtěli sledovat pouze druhou fázi třícestného podání ruky při navázání TCP komunikace, budeme chtít pakety s nastaveným příznakem SYN/ACK, tedy se dvěma bity. K tomu musíme použít tento příkaz:

```
tcp[tcpflag] & tcp-syn == 0x02
```

nebo

```
tcp[tcpflag] & tcp-syn == tcp-syn
```

Velikost paketu

Jazyk tcpdump filtru umožňuje pracovat i s metadaty – tedy s informacemi neobsaženými uvnitř paketů, ale s informacemi o nich. Velikost paketu je takto přístupná pod proměnnou len. Může být testována pomocí standardních aritmetických operátorů. Pro pakety menší než 100 bytů zadáme:

```
len < 100
```

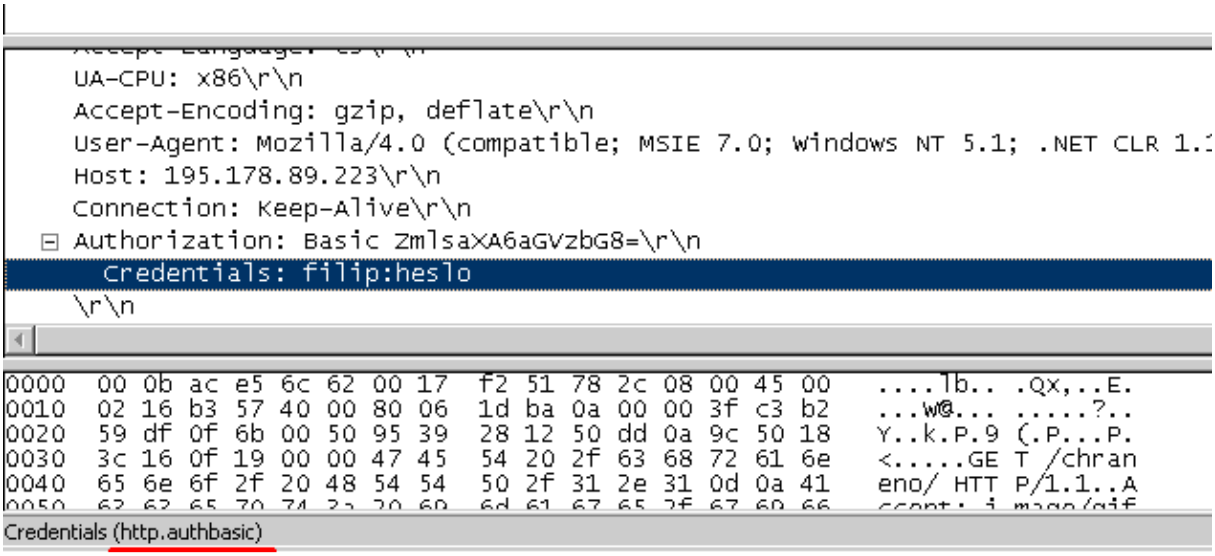
Zabudovány jsou také operátory less a greater – less je to samé jako „len <=“, greater jako „len >=“.

Příklady

- všechny HTTP pakety: tcp port 80
- ne-HTTP pakety: not tcp port 80 nebo !tcp port 80 nebo tcp port not 80 nebo tcp port !80
- HTTP provoz na stránku www.ethereal.com: tcp port 80 and dst www.ethereal.com
- HTTP provoz na všechny stránky kromě www.ethereal.com: tcp port 80 and not dst www.ethereal.com
- IPX pakety: ipx
- IPX pakety určené pro IPX síť 00:01:F0:EE: není možné, neboť z ipx paketů není možné extrahovat byty
- TCP pakety: tcp nebo ip proto 5
- TCP SYN pakety: tcp[tcpflag] & tcp-syn == tcp-syn
- IP pakety s celkovou délkou nad 255 bytů: ip[2:2] > 0xff
- IP nebo IPX pakety: ip or ipx

Zobrazovací filtry

Možnosti zobrazovacích filtrů se od zachycovacích filtrů liší – zachycovací filtr je totiž omezen knihovnou libcap na které je postavený. Zobrazovací filtry mají naopak k dispozici všechny položky, které jsou v rozbalovacím stromě ve středním panelu Wiresharku. Na rozdíl od zachycovacích filtrů používají tečkovou notaci. Seznam položek se kterými můžeme v zobrazovacích filtrech pracovat je vypsán v nápovědě (Help/Supported protocols/Display filter fields) ale nejjednodušší způsob jak zjistit výraz pro filtraci konkrétního protokolu nebo jeho pole je označit jej v nějakém zachyceném paketu ve středním panelu. Vlevo dole na stavové liště programu se pak ukáže název příslušného filtru, který by bylo nutné použít k vyfiltrování tohoto protokolu (pole).



Obrázek 17: Interaktivní nápověda k filtrovacím výrazům

Zobrazovací filtry jsou pomalejší než zachycovací filtry, na druhou stranu dokáží zpracovat téměř kterýkoliv protokol nebo pole protokolu. Pro otestování zda daný výraz funguje jako zobrazovací filtr jej stačí napsat do řádku Filter a to buď při živém zachycování paketů nebo při zobrazení paketů ze souboru.

Obrázek 18: Zadávání zobrazovacího filtru

K zobrazení všech IP paketů:

```
ip
```

nebo můžeme výběr omezit pouze na ty IP pakety které mají určité pole:

```
ip.len
```

Protože IP pakety mají vždy pole celková délka (len), je tento výraz ekvivalentní předchozímu. Nicméně jiné protokoly jako například TCP mají variabilní pole v hlavičce. Jedno z takových volitelných polí v TCP je MSS option, reprezentované hodnotou tcp.options.mss_val. Abychom našli všechny pakety které mají toto pole, zadáme dotaz:

```
tcp.options.mss_val
```

Zobrazovací filtry ve WireSharku jsou typové. To znamená, že v závislosti na typu pole může toto pole obsahovat pouze hodnoty daného typu. Typy hodnot pro zobrazovací filtry jsou tyto: Unsigned integer, Signed integer, Boolean, Frame number, Floating point, Double-precision, String - sekvence znaků, Byte string – sekvence hexadecimálních hodnot, Hardwarová adresa, IPv4, IPv6, IPX, síť, Absolutní čas, Relativní čas, Nic, Protokol.

Operátory, které mohou být použity k porovnávání hodnot jsou:

> nebo gt	větší než
>= nebo ge	větší nebo rovno
< nebo lt	menší než
<= nebo le	menší nebo rovno

== nebo eq	Rovno
!= nebo ne	Nerovno
contains	znak nebo řetězec v jiném řetězci
matches	regulární výrazy

Regulární výrazy pro operátor matches jsou převzaty ze skriptovacího jazyka PERL.
Příklady:

```
http contains "\"Yes\""
```

Hledá řetězec „Yes“ (včetně dvojitých uvozovek)

```
frame contains "\0777"
frame contains "\xff"
```

Hledá byty obsahující konkrétní hodnotu, v osmičkové a v šestnáctkové soustavě.

```
http contains "\\begin"
```

Hledá řetězec \begin (bez uvozovek ale s lomítkem)

```
eth.src == 00:09:f6:01:cc:b3
```

Hledá pakety s konkrétní MAC adresy

```
frame contains "POST"
```

Používá pseudoprotokol frame k nalezení všech paketů ve kterých je řetězec POST

```
frame contains 50:4f:53:54
http contains "GET"
http contains "User-Agent: Mozilla"
```

Adresy

```
ip.src == 192.168.1.1
ip.dst == wizard
ip.dst == www.ethereal.com
```

Hledá pakety, které mají zadané zdrojové nebo cílové IP adresy nebo DNS názvy

```
ip.addr == 192.168.1.0/24
ip.addr == wizard/24
```

To samé, využívá ale CIDR notaci

Čas

```
frame.time < "Dec 31, 2003 05:03:00"
```

Pakety které došly před 5:03 ráno silvestra 2003.

```
frame.time_delta > 0.02
```

Pakety které mezi sebou mají časovou mezeru větší než 0.02 sekundy

Rozsahy

```
eth.addr[0] == aa
eth[:2] == ff:ff
http[10:] contains 00:01:02
```

Podpůrné programy

Tshark (tetheral)

Konzolová verze Wiresharku ovládaná bez grafického rozhraní pouze přes příkazovou řádku. Jeho možnosti jsou víceméně stejné jako u grafické verze, výhodou je možnost zahrnutí vstupu a výstupu tohoto programu do vnějších skriptů obsahujících další programy.

Editcap

Slouží k odstranění paketů nebo k převodu formátu souborů, ve kterých jsou uložené pakety. Nepracuje přímo s naživo zachycenými pakety ale pouze ze soubory.

Mergecap
Kombinuje více souborů se zachycenými pakety do jediného souboru. Vstupní soubory můžou být v různých formátech.

Text2cap
Generuje soubory se zachycenými pakety z hexadecimálních výpisů síťového provozu. Do vytvořených souborů může vkládat uměle vytvořené hlavičky Ethernetu, IP, UDP a TCP protokolů.

Bezpečnost
Wireshark obsahuje obrovské množství dekodérů pro síťové protokoly. Počet vývojářů, kteří se na nich podíleli jde do stovek. Při takovém rozsahu práce je vždy riziko že některý z dekodérů může obsahovat chyby které mohou potenciálně představovat bezpečnostní riziko. V minulosti se několik takových chyb objevilo a v důsledku umožňovaly vzdálené převzetí kontroly nad systémem. Princip takového útoku spočíval v sestavení speciálního paketu obsahujícího sekvenci bytů, o které bylo známo že způsobí například přetečení zásobníku v dekodéru běžícího snifferu. Záludnost tohoto útoku je v tom že paket stačilo zaslat pouze do cílového segmentu sítě, nemusel být vůbec adresován přímo počítači na kterém běžel sniffer. Z tohoto důvodu byl Ethereal vyřazen z palety standardních nástrojů některých distribucí alternativních operačních systémů. Závěr: Wireshark je nezbytný nástroj pro práci bezpečnostních analytiků a správců sítě, je ale nutné mít vždy poslední záplatovanou verzi.

Úkoly pro cvičení
1. Stažení a instalace Wireshark do OS Windows běžící ve VMWare
2. Odposlech komunikace: SMTP, FTP, POP3, http, https, icq, případně další. Zhodnotit, které protokoly odposlouchávat jdou a které nikoliv. Pokusit se odposlechnout heslo při autentizaci na http stránky přes webové formuláře a přes http basic autentification.
3. Využít odposlechnutou komunikaci k „reverznímu inženýrství“ toho jak daný protokol (např. http,smtp, pop3, vzdálenou plochu, nntp, ..) pracuje a pak se pokusit provést stejnou činnost (stažení www stránky, odeslání emailu, výběr mailové schránky) z příkazové řádky pomocí program telnet nebo netcat.
4. Procvičit nejprve zachytávací a poté zobrazovací filtry ve Wiresharku, např:
   - Odfiltrovat všechen provoz směřující pouze na/z jedné konkrétní IP adresy (ne naší)
   - Odfiltrovat pouze http provoz
   - Odfiltrovat pouze http provoz učený na jednu IP adresu
   - Odfiltrovat pakety obsahujici urcite slovo www-authenticate: NTLM
   - Vynechat zároveň protokoly smb, udp, arp
   - Odposlechnout různé druhy www autentizace na webovém serveru IIS: anonymous, basic, integrated
   - Odfiltrovat provoz z jedné IP sítě
   - Odfiltrovat všechny http odpovědi s určitým kódem (např. 401, 404,..)
5. Vysledujte které pakety vysílá operační systém po zadání příkazů ipconfig /repase a následně ipconfig /renew

Úkoly pro seminární práce:

1. Odposlechnout provoz při zapínání operačního systému Windows (pouze provoz, který generuje samotný OS, neotevírat žádné www stránky apod.), formou seminární práce analyzovat všechy zachycené pakety – popsat jejich protokoly, jaký úkol plní daný protokol, proč je spuštěn při starut počítače apod.
2. Naprogramovat základní sniffer v jazyce C, C++ nebo C# (postačí bez filtrovacích funkcí, pouze základní zachytávání a zobrazení na obrazovku/do souboru)

4 Bezpečnost webových serverů

Webové servery jsou nejviditelnější částí Internetu, pro běžné uživatele prakticky představují Internet samotný. Na jejich bezpečnost a spolehlivost jsou kladeny mimořádně vysoké nároky. V minulosti trpěly mnoha neduhy a zranitelnostmi. V dnešní době jsou už při použití doporučených a vyzkoušených „best-practice“ postupů tyto problémy minulostí a zranitelné místo spíše představují webové aplikace na nich běžící. I tak je ale nutné mít o fungování a zabezpečení www serverů přehled.

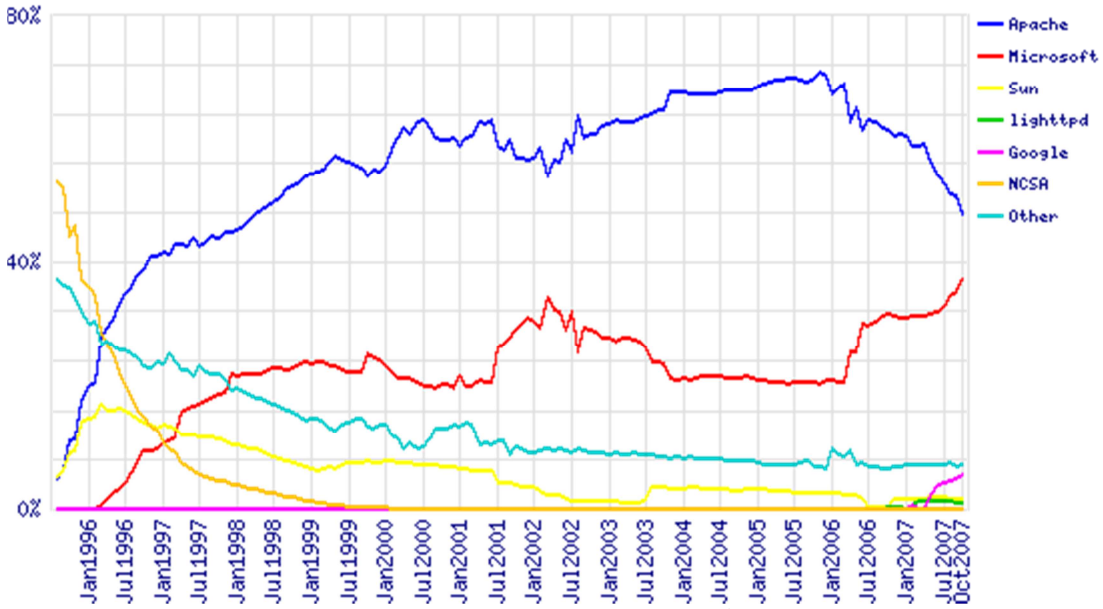
4.1 Webový server

Termín webový server může znamenat jednak počítač připojený k počítačové síti a poskytující www stránky nebo software nainstalovaný na tomto počítači. Webové servery patří k nejčastějším cílům počítačových útoků. Webový server (software) obvykle naslouchá na TCP portu číslo 80 (může se ale i lišit) na příchozí spojení a poté co je vytvořeno si s druhou stranou (klientem) vyměňuje data pomocí http (hyper text transfer protocol) protokolu. Principem této komunikace je v podstatě to že si klient vyžádá nějaký dokument dostupný na serveru (požadavek) a server mu odpoví buď, že dokument není dostupný nebo mu tento dokument pošle. Dokumentem mohou být www stránky, obrázky, soubory stylů CSS, skripty v JavaScriptu, Javě, Flashi a další obsah. Zdrojové kódy WWW stránek jsou napsány v html (hyper text markup language) jazyce. Součástí odpovědi je tzn. Stavový kód odpovědi. Ten udává, zda byl požadavek vyřízen v pořádku, či zda došlo k nějakým obtížím. Běžným stavovým kódem je označujícím stav OK je 200. Dále jsou to tyto skupiny kódů:

- 3xx - problémy spojené s přesměrováním
- 4xx - chyby související s vyřízením požadavku (stránka není dostupná, apod.)
- 5xx - interní chyby serveru

Server obvykle nějakým způsobem protokoluje vyřizované požadavky do tzn. logů. To umožňuje správci serveru vytvářet statistiky, optimalizovat obsah, hledat příčiny chyb případně sledovat pokusy o narušení bezpečnosti serveru.
Na www mohou být html soubory uloženy buď staticky přímo ve stejné podobě v jaké jsou zaslány klientovi, nebo mohou být dynamicky generovány až na základě přijatého požadavku. Statické dokumenty jsou přístupné mnohem rychleji, na druhou stranu dynamickými stránkami je možno reagovat na konkrétní požadavky. K dynamickému generování slouží například CGI (Common Gate Interface – rozhraní které umožňuje dynamicky generovat html obsah pomocí libovolného programovacího jazyka) skriptů, PHP (PHP: Hypertext preprocesor – rekurzivní zkratka), ASP.NET (Active Server Pages) případně pomocí dalších (oproti vypsaným méně rozšířených) technologií. Dynamicky generované stránky jsou často napojeny na databázovou vrstvu. Databáze jsou v podstatě tabulky skládající se ze sloupců (nazývaných atributy), řádků (záznamy) a z jazyka SQL (Structured Query Language), který nad těmito tabulkami pracuje (získávání a ukládání informací).
Nejrozšířenější programy, které poskytují www služby jsou:

- Apache HTTP Server
- Microsoft IIS Server
- Sun Java System Web Server



Obrázek 19: Vývoj tržního podílu webových serverů (zdroj netcraft.com)

4.1.1 Apache

Apache HTTP Server je v současnosti nejrozšířenější software pro poskytování www stránek. Je vyvíjen otevřenou skupinou vývojářů pod záštitou nadace Apache Server Foundation. Jeho počátky sahají až do roku 1994. Je dostupný pro celou řadu operačních systémů – Unix, FreeBSD, Linux, Solaris, Novell Netvare, Mac OS X a Microsoft Windows. Apache je zdarma dostupný software s volně dostupným zdrojovým kódem. V současné době (jaro 2008) je nasazený zhruba na 50% webových serverů Internetu. Domovská stránka projektu je http://httpd.apache.org/.

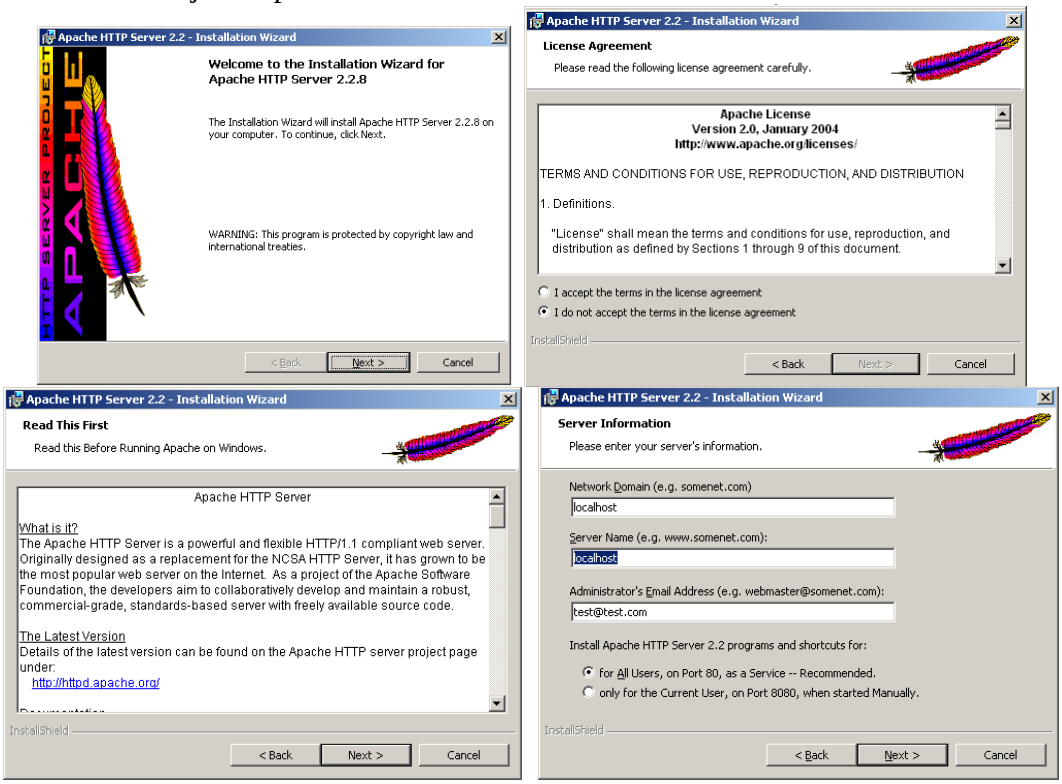
V současné době vedle sebe existují dvě paralelně vyvíjené verze – 1.3 a 2.2. Verze 2 byla oproti verzi 1 předělána od základů s ohledem na rozšiřitelnost. K Apache serveru je dostupná celá řada módů které rozšiřují základní funkcionalitu – mod_perl, mod_python, php (skriptovací jazyky), mod_access, mod_auth, mod_digest, mod_rewrite, proxy modul, mod_log, mod_zip a další. Apache podporuje virtuální hosting – jeden Apache web server může obsluhovat více www serverů najednou.
Velmi populární je kombinace open source řešení Apache + PHP skriptovací jazyk + MySQL databázový server.

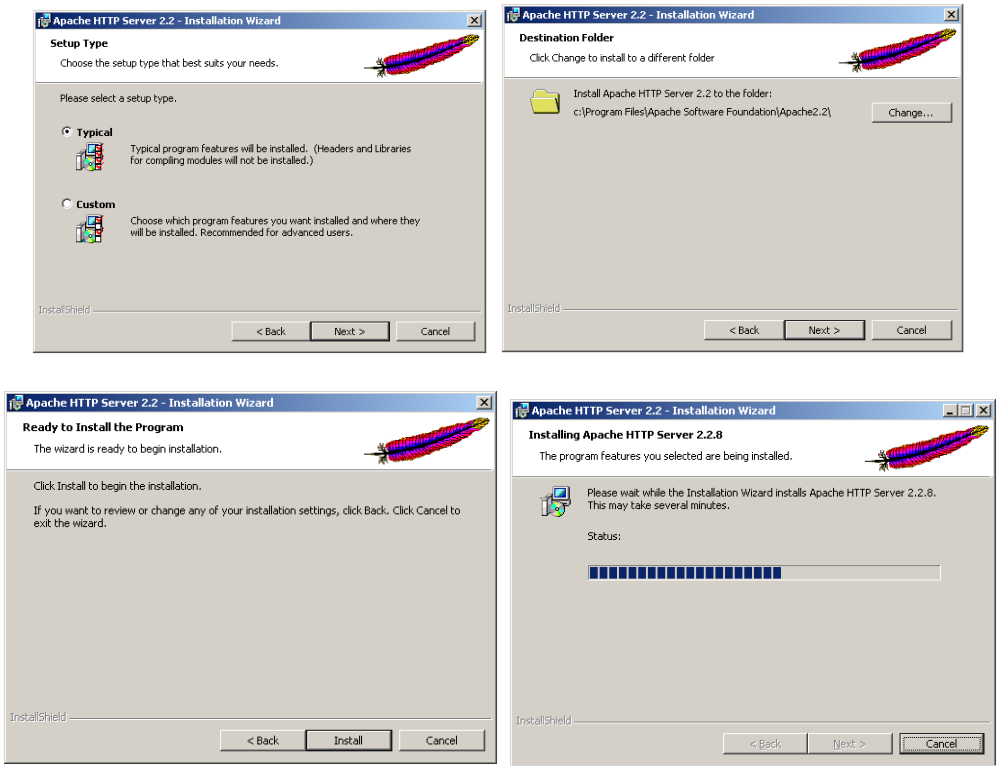
Instalace pod systémem Windows

Apache je zdarma ke stažení na adrese http://httpd.apache.org/, sekce Dowloads, verze označená Win32 Binary (MSI Installer). Odkazy ke stažení jsou volně dostupné, není potřeba registrace.

Obrázek 20: Stažení Apache http serveru

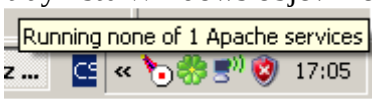
Instalaci odstartujeme spuštěním staženého exe souboru:





Obrázek 21: Instalace webového serveru Apache

Po dokončení instalace se dole v tray liště Windows objeví nová ikona:



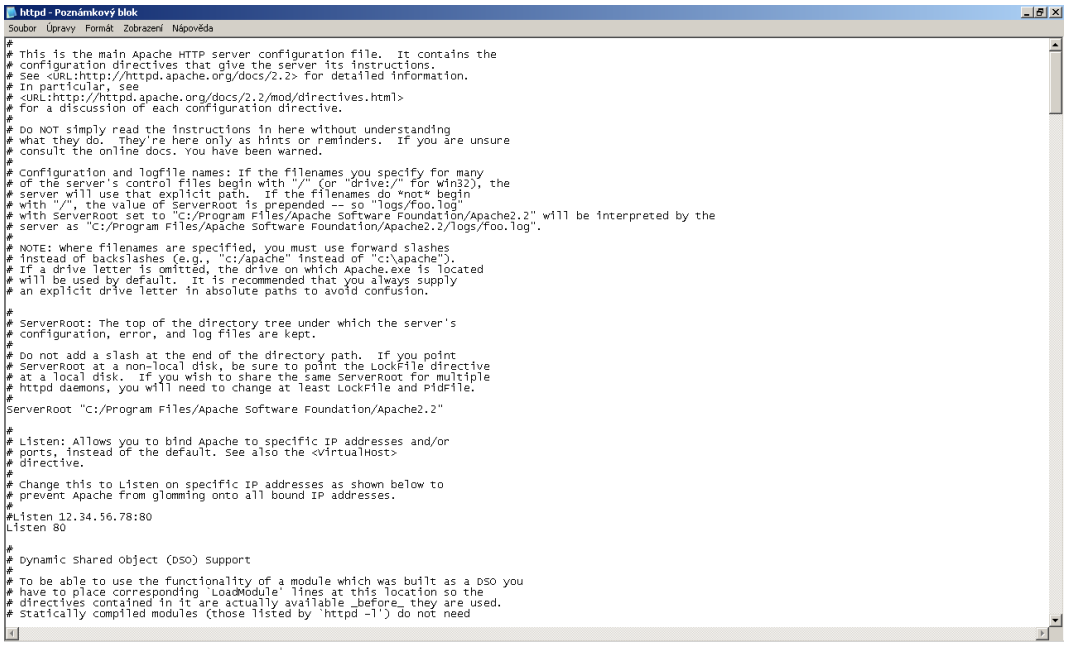
Obrázek 22: běžící Apache služba

V instalačním adresáři serveru je několik důležitých podadresářů:

/bin	- podpůrné programy od výrobce
/cgi-bin	- adresář pro CGI skripty
/conf	- konfigurační soubory, hlavně httpd.conf
/htdocs	- místo kam se nahrávají html (php) dokumenty
/modules	- kompilované moduly

Konfigurace Apache

Na běžné sestavě, na které neběží na portu 80 žádná jiná služba by se měl server rozběhnout ihned po instalaci bez problémů. V opačném případě je nutné upravit hlavní konfigurační soubor – httpd.conf. V tomto souboru se provádí veškeré globální nastavení serveru.



Obrázek 23: Soubor httpd.conf

Jedná se obyčejný textový soubor, editovaný v jakémkoliv základním editoru (např. notepad). Ty řádky, které v něm začínají znakem # jsou komentáře. Důležité direktivy jsou:
Kořenový adresář pro celý server (spustitelné soubory apod):
ServerRoot "C:/Program Files/Apache Software Foundation/Apache2.2"

Port, na kterém server běží:
Listen 80
Direktiva pro použití přídavného modulu:
LoadModule
Kořenový adresář pro www dokumenty:
DocumentRoot "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs"
Nastavení práv a chování pro jednotlivé adresáře:

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>
```

Po úspěšném nainstalování můžeme fungování apache ověřit přes webový prohlížeč zadáním adresy http://127.0.0.1 nebo http://localhost.

Omezení přístupu k adresáři v Apache

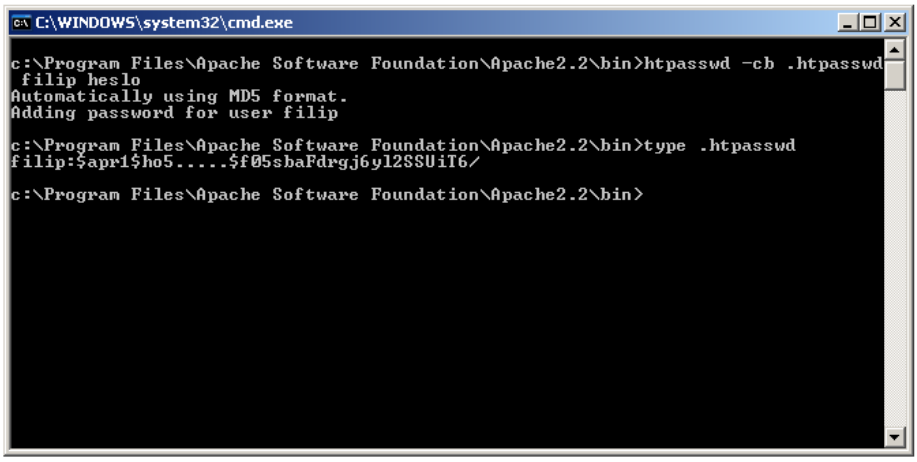
Adresáře jsou v Apache serveru chráněny pomocí nastavení direktiv v hlavním konfigurační souboru httpd.conf a pomocí souboru .htpasswd umístěném v dotyčném konkrétním adresáři. Nejprve ne nutné upravit direktivu (nebo vytvořit novou, pokud ji daný adresář ještě nemá) Directory v httpd.conf:

```
<Directory "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs/members">
        Options FollowSymLinks
        AuthType Basic
        AuthName "Tajná sekce"
```

```
        AuthUserFile "C:/Program Files/Apache Software
Foundation/Apache2.2/htdocs/members/.htpasswd"
        require valid-user
</Directory>
```

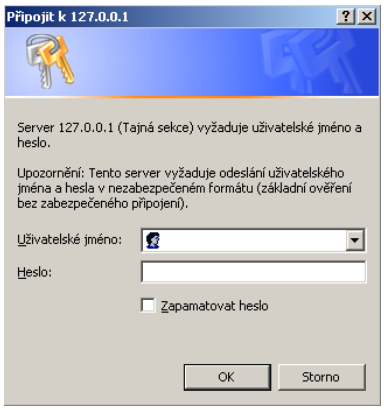
Upozornění - řádek AuthUserFile by na OS Linux používal tento tvar cesty : /usr/local/apache2/htdocs/members/.htpasswd

Kde jednotlivé řádky mají následující význam:
FollowSymLinks = lze v tomto adresáři používat symbolických linků dál na filesystém AuthType Basic = podporované jsou jen 2 možnosti: Basic – základní ověřování kdy jsou login a heslo posílány v čitelné podobě (ale zakódované pomocí base64), toto ověřování je nejpoužívanější a Digest – téměř nevyužívaný typ ověřování kdy se po síti neposílají loginy a hesla ale pouze jejich hash hodnoty. AuthName = text, který vám zobrazí prohlížeč v okně pro autentikaci AuthUserFile = soubor, vytvořený později příkazem "htpasswd", který obsahuje uživatelská jména a hesla pro přístup require valid-user = přístup k adresáři je na základě individuálního přihlašovacího jména a hesla
Dále je třeba vytvořit soubor .htpasswd a umístit jej do daného adresáře. V tomto souboru jsou umístěny loginy a hashe hesel pro přístup do adresáře. Tento soubor vytvoříme pomocí programu httpaswd který je v adresáři /bin. Při prvním spouštění programu je nutné zadat parametr –cb který vytvoří soubor s heslem. Při dalších spuštěních tento parametr již nezadávat, došlo by k přepsání souboru, zadáme pouze parametr –b, tím dojde k přidání uživatele do existujícího souboru.



Obrázek 24: Použití programu htpasswd

Například kombinace login: filip heslo: heslo vytvoří v .htpasswd následující řetězec:
filip:$apr1$MZ4.....$qRvvKy4cVJkqrmtRelA0G0
Zda je adresář opravdu zaveslovaný si ověříme v prohlížeči zadáním adresy příslušného adresáře na localhostu. Objeví se nám následující dialog:

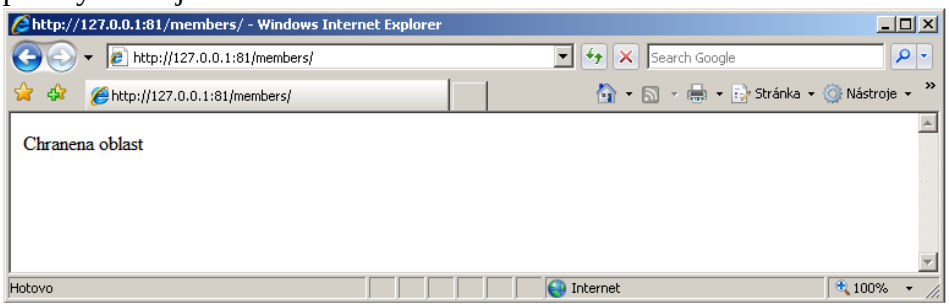


Obrázek 25: Dialog pro vyžádání loginu a hesla

Zadáním nesprávných údajů nám zobrazí pouze upozornění s hlavičkou 401:

Obrázek 26: Nesprávně zadané autentizační údaje

Zadáním správných údajů se dostaneme dále:



Obrázek 27: Správně zadané autentizační údaje

4.1.2 Internet Information Services

Výrobcem je Microsoft, hlavní část je HTTP/HTTPS server, kromě něj ale obsahuje i komponenty pro FTP, SMTP a NNTP služby. Po Apachi je to druhý nejpoužívanější server na Internetu, v současnosti (jaro 2008) má zhruba 40% podíl na trhu.

Verze:
IIS 1.0 – Windows NT 3.51, dostupný jako free add-on
IIS 2.0 – Windows NT 4.0
IIS 3.0 – Windows NT 4.0 Service Pack 3
IIS 4.0 – Windows NT 4.0 Option Pack
IIS 5.0 – Windows 2000
IIS 5.1 – Windows XP Professional
IIS 6.0 – Windows Server 2003, Win XP Pro x64 Edition
IIS 7.0 – Windows Vista, Windows Server 2008

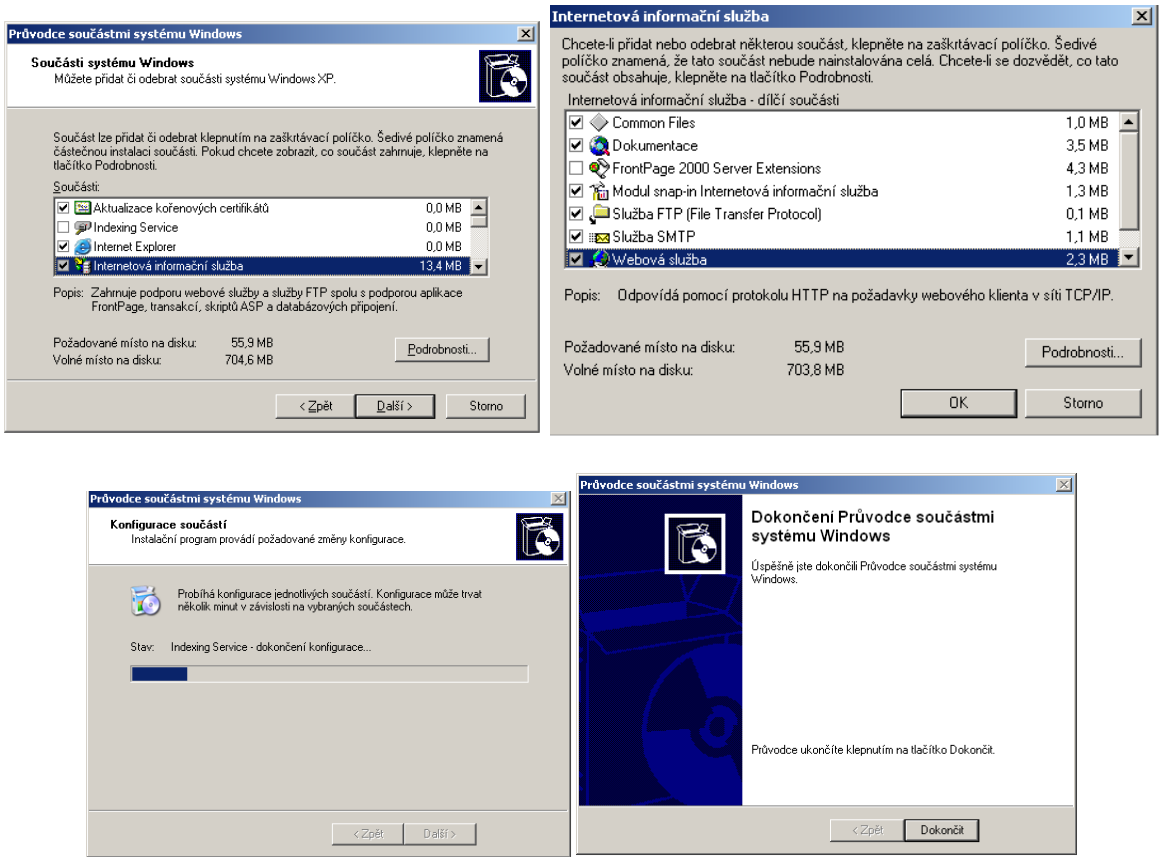
Windows XP Professional obsahuje ořezanou verzi IIS – dovoluje pouze 10 spojení naráz a jediné webové sídlo.

- Bezpečnost: dřívější verze obsahovaly řadu závažných bezpečnostních chyb, mimo jiné také neblaze proslulou CA-2001-19, která posloužila k šíření červa Code Red. V současné verzi 7.0 zatím není známý žádný exploit. Chybi nicméně nalezeny byly, není ale jisté zda je možné je zneužít (prezentováno na konferenci Hack in a Box 2008 v Dubaji, Cesar Cerrudo ze společnosti Argeniss).
- Ve verzi 5.1 a nižších byl IIS spouštěn s právy uživatele LocalSystem. Ve verzi 6.0 byly práva změněny na uživatele Network Service který má menší systémové privilegia. To v praxi znamená, že pokud by došlo k objevení a zneužití nového exploitu, nedošlo by ke kompromitaci celého operačního systému. Verze 6.0 také obsahuje přepsané jádro http zásobníku (http.sys) s přísnějším parserem http požadavků.
- Verze 7.0 je od základu vytvořena modulárně (dříve byl program „monolitický“). Základem je malé jádro web serveru, další moduly mohou být přidávány podle toho zda jsou v konkrétní situaci potřebné. Mezi moduly dodávané „z krabice“ patří http, bezpečnostní, obsahový, kompresní, cachovací, logovací a diagnostický modul.
- Další významnou změnou je že konfigurační soubory jsou nyní uchovávány jako xml soubory místo v metabázi. Server má globální konfigurační soubor který poskytuje výchozí nastavení. Každý adresář může obsahovat vlastní web.config soubor který pro něj překryje globální nastavení. Na rozdíl od předchozích verzích se změna v těchto souborech projeví okamžitě.

Možnosti řízení přístupu k chráněným dokumentům:
- basic autentifikace
- digest autentifikace
- integrovaná autentifikace systému Windows
- .NET passport autentifikace

Podobně jako Apache je i IIS od novějších verzí předěláván pro větší podporu modularity. Od verze IIS 7.0 je monolitické jádro nahrazeno moduly. Následující moduly jsou dodávány v základní instalaci:
- http modul
- bezpečnostní modul
- obsahový modul
- kompresní modul
- caching modul
- logovací a diagnostický modul

Instalace IIS na Windows XP Pro
Internet Information Services je součástí mnoha verzí operačního systému Windows. Jednou z nich je také Windows XP Pro (ve verzi Home není i když existuje výrobcem nepodporovaný způsob jak jej rozjet), ve které však není nainstalovaný defaultně z instalace. Pro nainstalování jsou nutné následující kroky:
Přejít do Start/Nastavení/Ovládací panely/Součásti systému Windows, otevře se „Průvodce součástmi systému Windows“, dále zaškrtnout položku „Internetová informační služba“ (může vyžadovat instalační cd).:

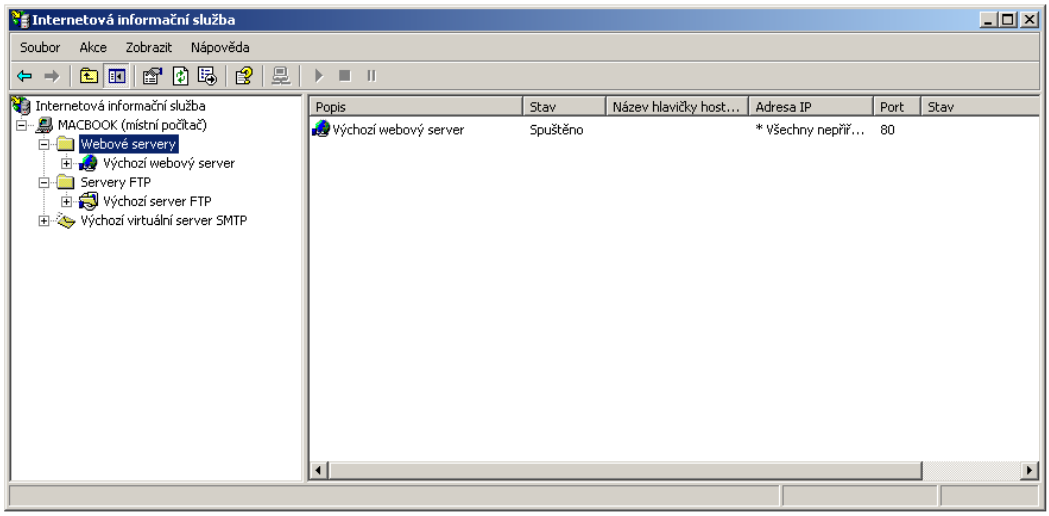


Obrázek 28: Instalace IIS

Po instalaci je vytvořena složka c:\inetpub\www, výchozí složka pro html a asp.net stránky. Funkčnost serveru můžeme vyzkoušet v prohlížeči zadáním adresy http://localhost nebo http://127.0.0.1.

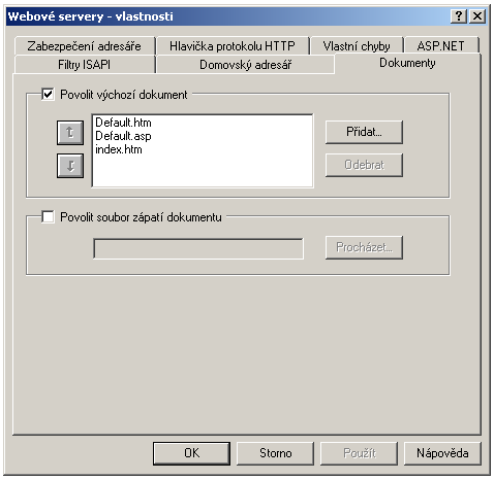
Správa IIS

Administrace IIS se provádí přes takzvaný console snap-in modul, který se nachází ve složce %SystemRoot%\system32\inetsrv\iis.msc. Pro jeho pohodlné spouštění je třeba vytvořit zástupce na tento soubor.



Obrázek 29: Konzole pro správu IIS

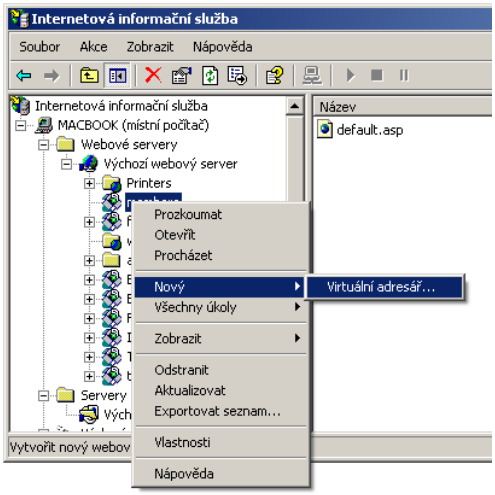
V tomto grafickém rozhraní je nalevo rozbalovací struktura, ve které jsou přístupné všechny servery a v nich všechny soubory a aplikace. Nastavení každé položky (server, aplikace, virtuální adresář, soubor) se provádí přes pravé tlačítko myši a položku vlastnosti, např. vlastnosti webového serveru:



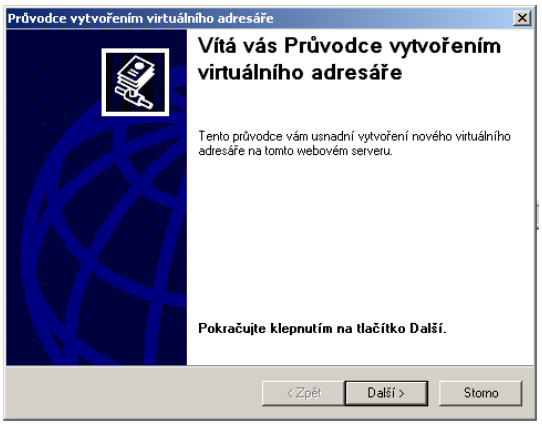
Obrázek 30: Vlastnosti webového serveru

Omezení přístupu k adresářům v IIS

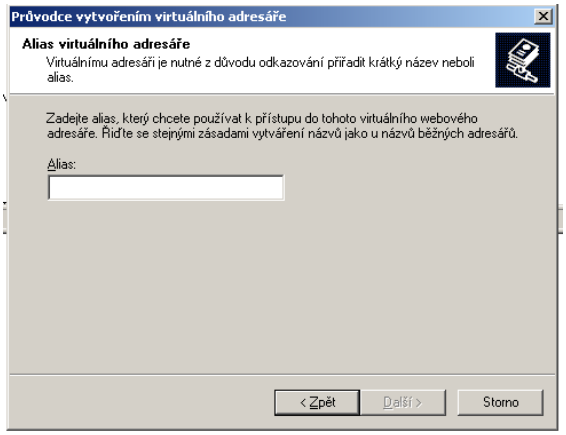
Nejprve vytvoříme nový virtuální adresář:



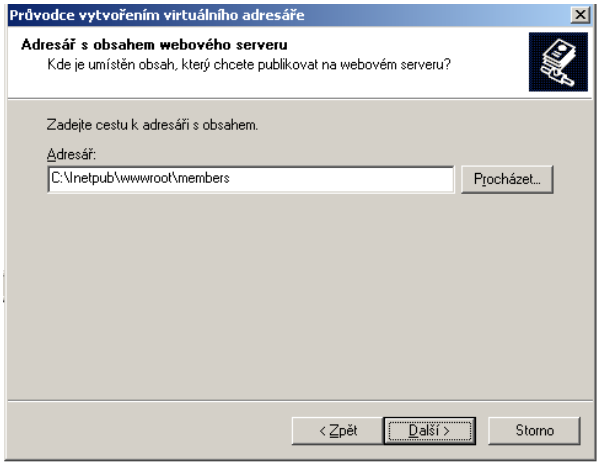
Obrázek 31: Vytvoření virtuálního adresáře



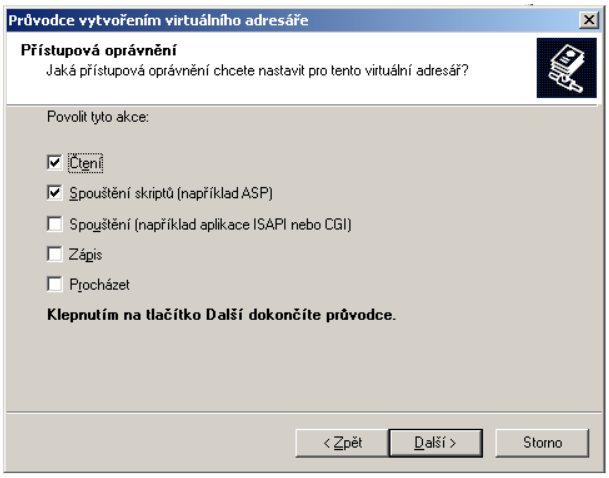
Obrázek 32: Průvodce vytvořením virtuálního adresáře



Obrázek 33: Název virtuálního adresáře (alias)

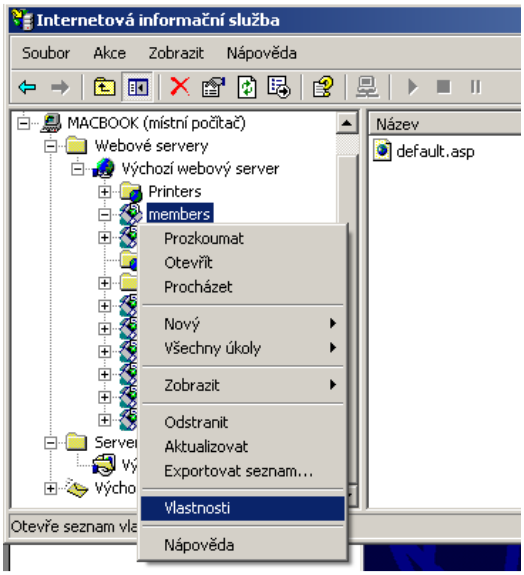


Obrázek 34: Přiřazení skutečného adresáře virtuálnímu

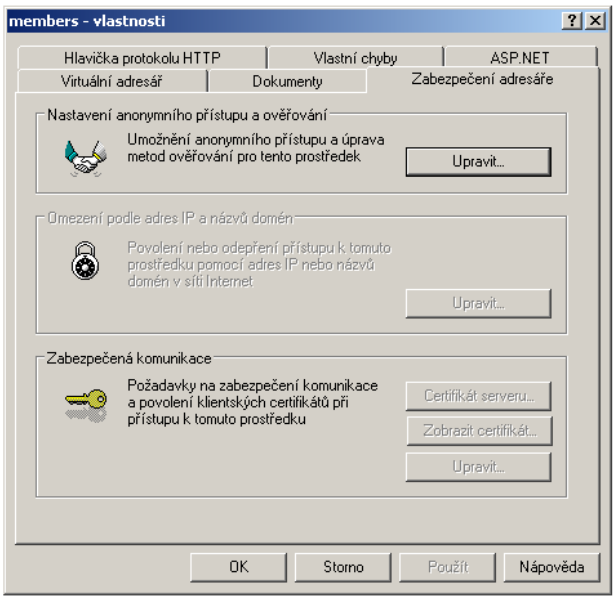


Obrázek 35: Nastavení práv pro virtuální adresář

Na poslední obrazovce pouze potvrdíme dokončení akce. Tím je vytvořen virtuální adresář (tj. ten který se používá ve www adrese). Nyní je potřeba ve vlastnostech tohoto adresáře nastavit zabezpečení:

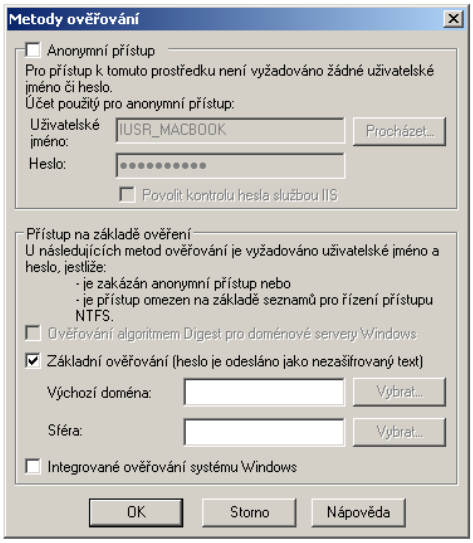


Obrázek 36: Přístup k vlastnostem virtuálního adresáře



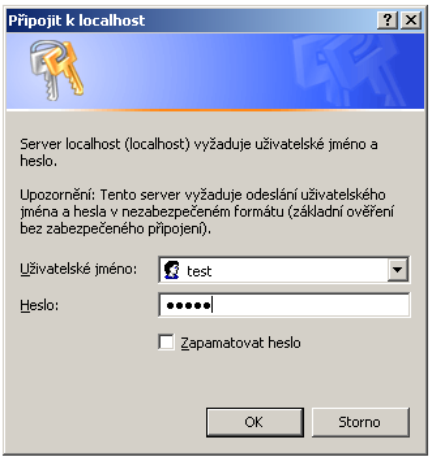
Obrázek 37: Jednotlivé položky vlastností virtuálního adresáře

V dialogu Metody ověřování je nutné odškrtnout volby „Anonymní přístup“, „Integrované ověřování systému Windows“ a zaškrtnout „Základní ověřování“.



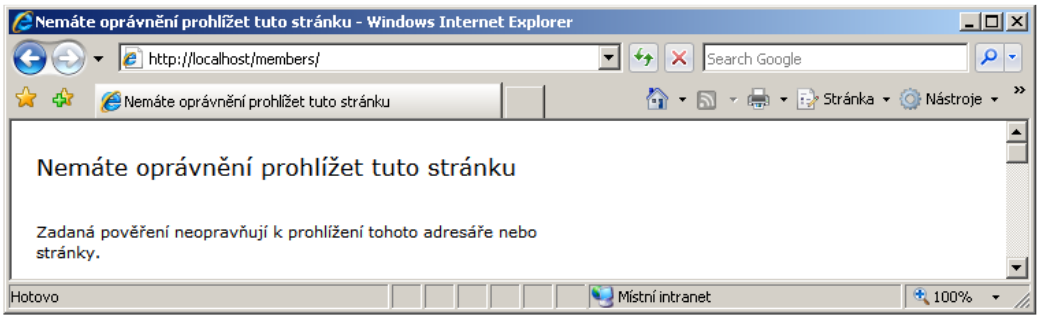
Obrázek 38: Dialog Metody ověřování

Jako login a heslo budou nyní použitelné název a heslo z účtu kteréhokoliv uživatele naší instalace Windows. Nyní je adresář http://localhost/members/ chráněn a při pokusu se do něj dostat je vyžadován login a heslo:



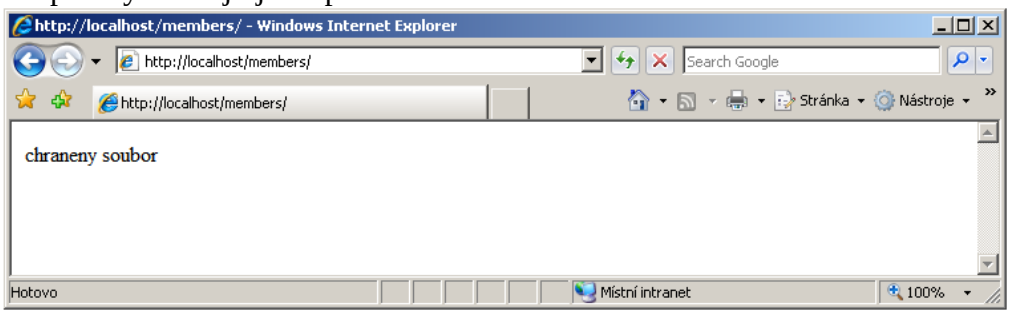
Obrázek 39: Dialog pro zadání hesla ve webovém prohlížeči

Při špatném zadání hesla se dostaneme pouze na stránku s upozorněním:



Obrázek 40: Nesprávně zadané údaje

Při zadání správných údajů jsme puštěni k chráněnému obsahu:



Obrázek 41: Správně zadané údaje

Do tohoto adresáře můžeme přistupovat tak dlouho, dokud nedojde k vypršení platnosti sezení (například pokud stránku otevřeme v jiné instanci prohlížeče).

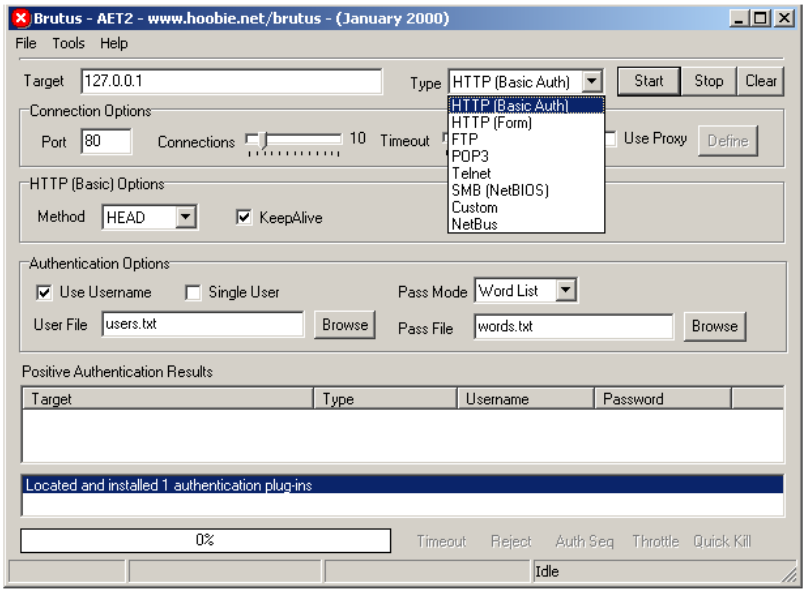
4.1.3 Získání neoprávněného přístupu do adresářů s omezeným přístupem

Možností je několik – získat login a heslo například trojským koněm umístěným v počítači uživatele, odposlechem síťového provozu v místním segmentu sítě nebo slovníkovým útokem. Pro první dva případy je ale nutný fyzický přístup k danému počítači nebo místní síti, případně zneužití zranitelností vyšších úrovní a získání vzdáleného přístupu. Čistě odkudkoliv z Internetu nám bez jiných zranitelných míst systému nezbývá nic jiného než zkusit slovníkový útok – v podstatě zkoušíme jednu kombinaci loginu a hesla za druhou dokud nás to nepřestane bavit, dokud se netrefíme do správné kombinace nebo dokud se s námi server neodmítne bavit s naší IP adresou což dobře nastavený server udělá již po 3-5 nesprávných

pokusech. V tom případě musíme využít připojení přes proxy servery. Ručně by se jednalo o velmi zdlouhavou práci měnit proxy server po každých pár pokusech ale i k tomu úkolu jsou naštěstí k dispozici automatické nástroje.

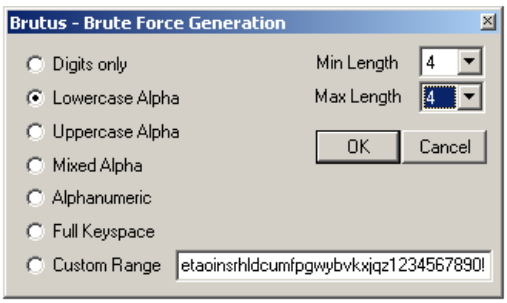
4.1.4 Brutus AET2

Jeden z nejznámějších nástrojů pro slovníkový útok. Dokáže zkoušet hesla na širokou řadu služeb – http (basic), http (formuláře), ftp, pop3, telnet, NetBIOS, NetBus (starší trojský kůň). Je modulární – podpora jiných protokolů jde dopsat pomocí speciálních plugin souborů. Program je zdarma ke stažení na adrese http://www.hoobie.net/brutus/brutus-download.html. Po stažení jej stačí pouze rozbalit z archivu, neinstaluje se. Po spuštění se ukáže následující okno:



Obrázek 42: Brutus AET2

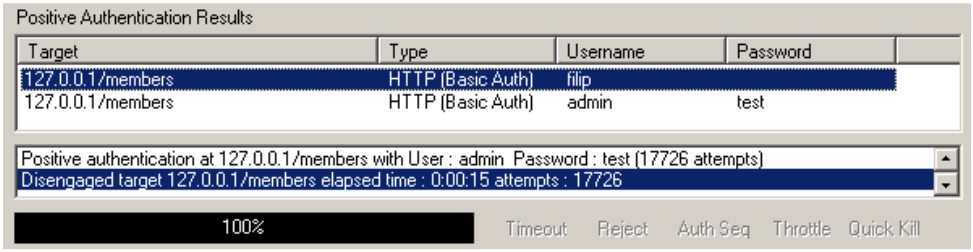
K otestování námi nastavených www serverů vybereme volbu http basic auth. Uživatelské jméno je možné buď zadat jedno napevno nebo určit textový soubor ze kterého budou vyzkoušena všechna jméno (formát souboru je jedno jméno na řádek). Stejně tak hesla mohou být zkoušena ze souboru (opět jedno heslo na řádek) nebo vytvořena kombinatoricky – lze vybrat minimální délku hesla, maximální délku hesla a abecedu ze které budou hesla tvořeny. K tomu aby byl Brutus schopný správně odhalit neúspěšný pokus o nalogování, musí server vracet standardní hlavičku 401.



Obrázek 43: Možnosti nastavení kombinatorického útoku

Následující screenshot ilustruje jeden falešně pozitivní (první řádek, jednalo se o IIS server který při negativní autentizaci nevracel standardní hlavičku 401, tím pádem byl neúspěšná

autentizace považována za úspěšnou – pro ošetření by bylo potřeba vytvořit speciální plugin) a jeden pravý případ nalezení kombinace loginu a hesla.



Obrázek 44: Výsledky testu

4.1.5 Zabezpečení Internet Information Services

Pro důkladné zabezpečení serveru se službou IIS je nutné provést bezpečnostní opatření v následujících částech systému:

- síťové prostředí
- patche a updaty
- windows services
- uživatelské účty a skupiny
- souborový systém
- auditování a logování

4.1.6 Síťové prostředí

Zajištění bezpečného síťového prostředí je základní podmínka. Potřebujeme následující nástroje:

- router
- firewall
- intrusion detection system

Router je buď softwarová aplikace nebo hardwarové zařízení zodpovědné za forwardování IP paketů mezi sousedícími sítěmi. Typický router má možnost nastavit tzv. ACL – Access Control List neboli přístupový seznam ve kterém jsou specifikovány toky paketů mezi vnitřní a vnější sítí. ACL lze použít k zablokování nebo filtrování nechtěných paketů (ve smyslu nechtěných IP adres nebo jejich rozsahů). Při instalaci nového serveru je nutné natavit ACL na routeru tak aby před dokončením instalace a všech následných zabezpečovacích kroků nebyl www server přístupný z vnější sítě. Pokud jsme v organizaci, která si router spravuje sama můžeme to udělat sami, pokud máme připojení spravované ISP musíme požádat jej.
Firewall je opět softwarové nebo hardwarové zařízení které filtruje jeden nebo více protokolů. Windows od verze 2003 obsahují základní softwarový firewall Internet Connection Firewall (ICF). Defaultně je tato funkce vypnuta. Po jejím zapnutí můžeme nakonfigurovat, které aplikace v počítači mají povoleno navazovat spojení ven na síť (na kterémkoliv portu potřebují) nebo které porty jsou otevřeny na naslouchání (tzn. jsou otevřeny pro počítače venku na síti které se na ně snaží připojit). K zajištění fungování jednotlivých funkcí IIS serveru musíme zvážit povolení následujících portů:

Hypertext Transfer Protocol (HTTP)	80
http secure (HTTPS)	443
File Transfer Protocol (FTP) příkazový kanál	21
FTP datový kanál	22
Simple Mail Transfer Protocol (SMTP)	25

Post Office Protocol v3 (POP3)	110
Network News Transfer Protocol (NNTP)	119
Secure NNTP	563

Firma Microsoft doporučuje použít ICF pro malé a středně velké webové projekty. Pro větší projekty je lepší použít sofistikovanější nástroje (např. Microsoft Internet Security and Acceleration (ISA) server nebo jiný komerční Firewall, např. od firem CheckPoint, Cisco).
Intrusion Detection System je síťové zařízení (většinou software) které zachytává a analyzuje síťový provoz s cílem identifikovat potenciálně útočné aktivity. Primárně je určeno pro detekci ale dokáže také případně do síťového provozu zasáhnout a ukončit nebezpečné spojení. IDS dokáže detekovat jinak nezjistitelné útoky jako Denial of Service, IP spoofing, session hijacking.
Doporučení pro zabezpečení síťové infrastruktury v oblasti síťového zabezpečení počítače s www serverem:

- filtrovat provoz podle protokolů a portů. Pokud například provozujeme pouze www server, povolíme ve firewallu pouze porty 80 a 443, ostatní zakážeme
- zakázat požadavky (requests) protokolu Internet Control Message Protol (ICMP). Útočníci poté nebudou schopní použít na náš server standardní ping, tím jim jednak ztížíme nalezení serveru, jednak zabráníme útokům spojeným s protokolem ping (ping of death, ping flood)
- zakázat NetBIOS nad TCP/IP. Služby NetBIOS pak nebudou moci využívat TCP port 139 a nebude možné navázat potenciálně nebezpečené NetBIOS session
- zavést u síťových zařízení logování a auditování logů v pravidelných intervalech
- omezit fyzický a vzdálený přístup k síťovým zařízením

4.1.7 Patche a Updaty

Bezpečnostní záplaty, opravy, servis packy a updaty jsou kritickou součástí zabezpečení operačního systému a IIS. Nezáplatovaný operační systém ve výchozí instalaci je snad v každé verzi operačního systému Windows zranitelný některou ze známých chyb. Používání záplat je tedy bezpodmínečně nutné. Na druhou stranu ale i záplaty samotné mohou někdy obsahovat chyby nebo jinak negativně ovlivnit běh aplikací, proto je nutné před jejich nasazením na živou produkční aplikaci je vyzkoušet na testovacím stroji. Na svůj stroj aplikujeme samozřejmě pouze ty patche které potřebujeme – pokud nepoužíváme SQL server je zbytečné stahovat a instalovat pro něj opravy. Z hlediska IIS je nutné mít záplatovány především tento software:

- Microsoft Windows: patche, updaty a service packy pro operační systém
- Microsoft Data Access Component (MDAC): tato komponenta je různými aplikacemi používána pro přístup k databázovým severům a jiným datovým zdrojům
- Windows Scripting Engine: engine pro jazyky VBScript a JScript je využíván staršími Active Server Pages (ASP)
- .NET framework: rozhraní využívané ASP.NET stránkami

O tom jaké záplaty a updaty potřebujeme se můžeme dozvědět na následujících místech:
Security Bulletin: nejaktuálnější informace o bezpečnosti produktů Microsoft jsou na adrese www.microsoft.com/security/. Bezpečnostní bulletiny je také možné odebírat emailem. V bulletinu jsou vždy k dispozici technické informace o problému, popsáno řešení a seznam postižených produktů a jejich verzí.
Windows Automatic Updates: tato aplikace je součástí OS Windows, automaticky hlídá, stahuje a podle nastavení i instaluje záplaty a updaty. Výhodou je že se správce o patche a

updaty nemusí starat – v produkčním prostředí se nicméně tato možnost většinou nevyužívá protože patche a updaty je nutné nejprve ověřit na záložních strojích.
Windows Updates: služba postavená na stejném systému jako WAU, ale dostupná na internetu na adrese http://windowsupdate.microsoft.com. Při návštěvě této stránky je počítač automaticky skenován na potřebné záplaty.
Microsoft BaseLine Security Analyzer (MBSA): tato aplikace je schopná detekovat patche a updaty nainstalované nejen na lokálním počítači ale také na okolních počítačích v síti. Nástroj je volně ke stažení na stránkách www.microsoft.com
Doporučení pro patche a updaty:

- pravidelně kontrolovat a aplikovat nejnovější patche a updaty
- stahovat je pouze z důvěryhodných zdrojů (microsoft.com, stránky výrobce software)
- nepoužívat updaty nebo podobně se tvářící soubory došlé emailem
- před nasazením na produkční stroj testovat patche na zkušebních serverech
- mít přehled o změnách a důsledcích které nastanou v systému aplikováním patchů

4.1.8 Windows Services

Služby systému Windows jsou programy, které běží na pozadí systému a poskytují uživatelům a ostatním programům nějakou funkcionalitu. Přehled o tom jaké služby běží na našem systému lze získat spuštěním konzole services.msc (ve Windows XP Start/Nastavení/Ovládací panely/Nástroje pro správu/Služby, ve Windows 2003 Start/Nástroje pro správu/Služby). Služby lze v tomto nástroji nastavovat, spouštět, zastavovat, zakazovat a povolovat. Z hlediska minimalizování útočeného prostoru je dobré mít v systému pouze ty služby které potřebujeme.
Doporučení pro služby:

- neinstalovat nepotřebné komponenty a služby
- zastavit a zakázat nevyužívané služby
- neinstalovat aplikace třetích stran pokud je nepotřebujeme

Uživatelské účty a skupiny
Doporučení pro účty:

- odstranit nepoužívané účty
- pokud je u nějakého účtu známo že se nebude po určitou dobu využívat, disablovat jej
- zakázat účet Guest (host)
- přejmenovat účet Administrator
- vynucovat politiku silných hesel
- používat politiku uzamčení účtů po několika neúspěšných pokusech o přihlášení
- zalogovávat se do systému s nižšími privilegii než Administrator
- zakázat null sessions

Doporučení pro souborový systém
Na webovém serveru je naprosto nezbytné využívat souborový systém NTFS. Z hlediska bezpečnosti proti selhání hardwaru se obvykle www servery nasazují na RAID pole.

4.1.9 Logování a kontrola logů (auditing)

Logování zaznamenává události které na serveru nastaly do příslušného souboru. Auditovaném těchto souborů se snažíme najít neobvyklou aktivitu. Logování a kontrola logů sice nezabrání prvnímu útoku ale pokud jej dokážeme identifikovat umožní nám bránit se v budoucnosti a pochopit slabiny našeho systému. Je důležité soubory s logy dostatečně zabezpečit protože zkušení útočníci se pokusí je smazat nebo změnit aby zametli stopy. Následující zdroje logů jsou důležité pro správce IIS:

- Event Viewer: zaznamenává události v operačním systému a v IIS aplikaci. Tyto logy se dělí do tří hlavních skupin: system, security a application.
- IIS Site Activity: log který sleduje klientské požadavky – kdo, kdy, odkud a jaký obsah si vyžádal
- HTTP API Error: tyto logy jsou generovány kernel-mode ovladačem HTTP.SYS. Tato nízkoúrovňová komponenta operačního systému zachytává HTTP požadavky ze síťového rozhraní a přesměrovává je do příslušného apllication poolu.
- URL Scan: Internet Server API (ISAPI) filtr který je volitelnou součástí IIS 6.0. Monitoruje HTTP požadavky na základě určených pravidel. Pokud požadavek tyto pravidla nesplňuje, vrátí server chybovou stránku „404 File Not Found“.

Doporučení pro logování a kontrolu:
- Logovat neúspěšné pokusy o přístup k prostředkům chráněným heslem (složka, soubor, aplikace)
- Zapnout IIS site activity logging
- Přemístit defaultní IIS logovací soubor a zabezpečit ho NTFS právy
- Zálohovat a archivovat staré logy
- Logovat neúspěšné pokusy o přístup k důležitým systémovým souborům
- Pravidelně kontrolovat soubory s logy

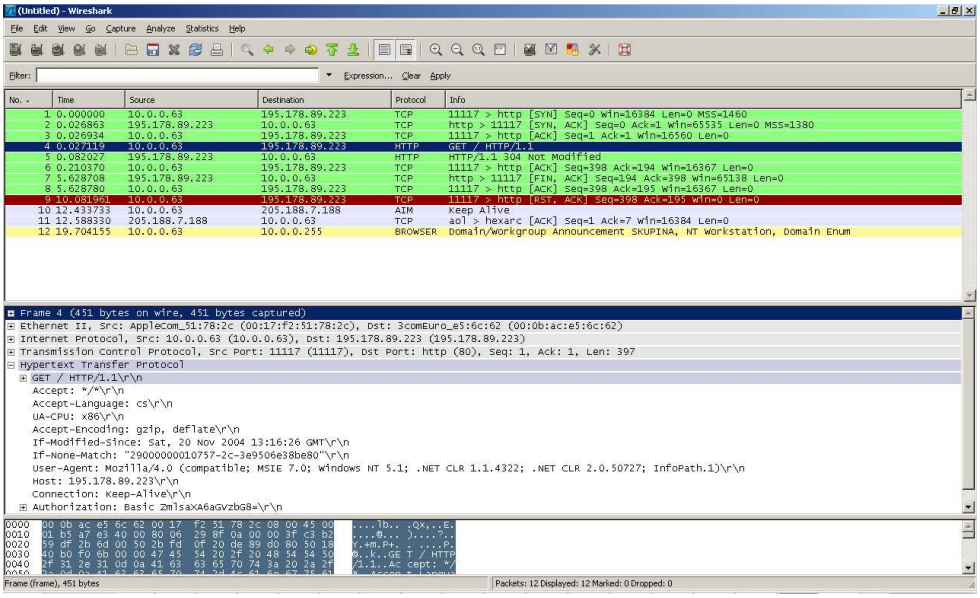
4.1.10 HTTP - Hypertext Transfer Protocol

Protokol, který se stará o výměnu dat mezi www serverem a www prohlížečem. Funguje na principu požadavek – odpověď. Uživatel pošle serveru dotaz ve formě čistého textu (nejčastěji pomocí prohlížeče, ale lze například i přes příkazovou řádku programy jako telnet, netcat). Dotaz obsahuje umístění požadovaného dokumentu, informace o schopnostech prohlížeče, autentizační údaje (jsou-li vyžadovány), informace přečtené ze souboru cookie apod. Server vrátí odpověď opět ve formě několika řádek čistého textu popisujícího výsledek dotazu (zda se dokument podařilo najít, jakého je typu, zda a čím je komprimovaný,..) za kterými následují samotné data dokumentu.
Pokud uživatel bude mít po chvíli další dotaz na stejný server (např. proto, že uživatel v dokumentu kliknul na hypertextový odkaz), bude se jednat o další, nezávislý dotaz a odpověď. Z hlediska serveru nelze poznat, jestli tento druhý dotaz jakkoli souvisí s předchozím. Kvůli této vlastnosti se protokolu HTTP říká bezestavový protokol – protokol neumí uchovávat stav komunikace, dotazy spolu nemají souvislost. Tato vlastnost je nepříjemná pro implementaci složitějších procesů přes HTTP (např. internetový obchod potřebuje uchovávat informaci o identitě zákazníka, o obsahu jeho „nákupního košíku“ apod.). K tomuto účelu byl protokol HTTP rozšířen o o tzv. http cookies, které umožňují serveru uchovávat si informace o stavu spojení na počítači uživatele. Jako cookie se v protokolu HTTP označuje malé množství dat, která WWW server pošle prohlížeči, který je uloží na počítači uživatele. Při každé další návštěvě téhož serveru pak prohlížeč tato data posílá zpět serveru. Cookies běžně slouží k rozlišování jednotlivých uživatelů, ukládá se do nich obsah „nákupního košíku“ v elektronických obchodech, uživatelské předvolby apod. Fyzicky mají cookie podobu textového souboru. Cookies neznamenají žádné nebezpečí pro počítač jako takový. Přesto cookies mohou být nebezpečné pro ochranu soukromí. Navštívený web si totiž může ukládat do cookies jakékoliv informace, které o návštěvníkovi zjistí a může tak postupně zjišťovat zájmy konkrétního návštěvníka. Které stránky navštěvuje, jaké informace vyhledává, jak často daný web navštěvuje apod.Těchto informací se dá posléze i proti vůli návštěvníka využívat pro cílenou reklamu, statistické vyhodnocování chování návštěvníků, apod.

Ukázka příkazu
Klient zašle dotaz:
GET /wiki/Wikipedie HTTP/1.1
Host: cs.wikipedia.org
User-Agent: Mozilla/5.0 Gecko/20040803 Firefox/0.9.3
Accept-Charset: UTF-8,*
Server odpoví:
HTTP/1.0 200 OK
Date: Fri, 15 Oct 2004 08:20:25 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.8
X-Powered-By: PHP/4.3.8
Vary: Accept-Encoding,Cookie
Cache-Control: private, s-maxage=0, max-age=0, must-revalidate
Content-Language: cs
Content-Type: text/html; charset=utf-8

Za touto hlavičkou následuje jeden prázdný řádek (označující její konec) a pak požadovaný HTML dokument. Hlavička obsahuje informaci o tom, že dotaz se podařil (první řádek: „200 OK“), datum a čas vyřízení dotazu, popis serveru, který odpovídá, informace o typu vráceného dokumentu (MIME typ text/html v kódování UTF-8) a další informace.

Pokud použijeme program jako například Ethereal (Wireshark), můžeme zachytit packety ze sítě a sami se podívat jaké přesně dotazy náš prohlížeč vysílá:

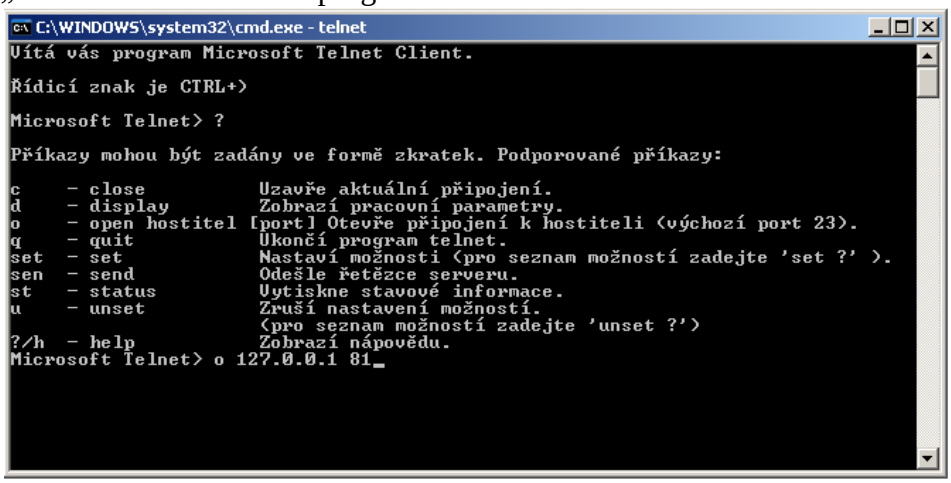


Obrázek 45: Zachycená http komunikace

Dotazovací příkazy
HTTP definuje několik metod, které se mají provést nad uvedeným objektem (dokumentem). Syntaxe dotazů je: <metoda> <objekt> HTTP/<verze>
GET
Požadavek na uvedený objekt. Je to nejpoužívanější metoda. Je používána když se podíváte ráno na zpravodajský i-magazín, přes den stahujete RSS a nebo stahujete novou verzi webového prohlížeče z webu.
HEAD

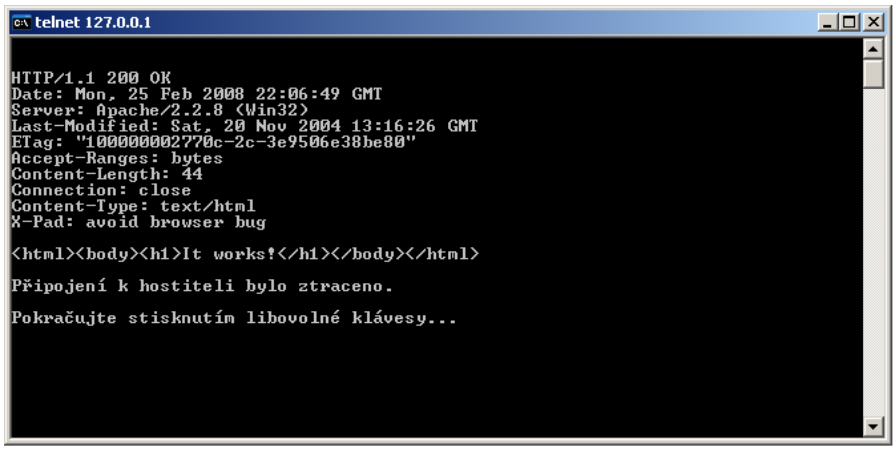
To samé jako metoda GET, ale už nepředává data. Poskytne pouze metadata o požadovaném cíli (velikost, typ, datum změny, …).
POST
Odesílá uživatelská data na server. Používá se například při odesílání formuláře na webu. S předaným objektem se pak zachází podobně jako při metodě GET.
PUT
Nahraje data na server. Objekt je jméno vytvářeného souboru. Používá se velmi zřídka, pro nahrávání dat na server se běžně používá FTP nebo SCP/SSH.
DELETE
Smaže uvedený objekt ze serveru. Jsou na to potřeba jistá oprávnění stejně jako u metody PUT.
TRACE
Odešle kopii obdrženého požadavku zpět odesílateli, takže klient může zjistit, co na požadavku mění nebo přidávají servery, kterými požadavek prochází.
OPTIONS
Dotaz na server, jaké podporuje metody.
CONNECT
Spojí se s uvedeným objektem před uvedený port. Používá se při průchodu skrze proxy pro ustanovení kanálu SSL

Telnet
Otestovat fungování http protokolu můžeme například pomocí telnetu. V příkazovém řádku napíšeme „telnet“. Otevře se okno programu Telnet Microsoft Klient.



Obrázek 46: titulek

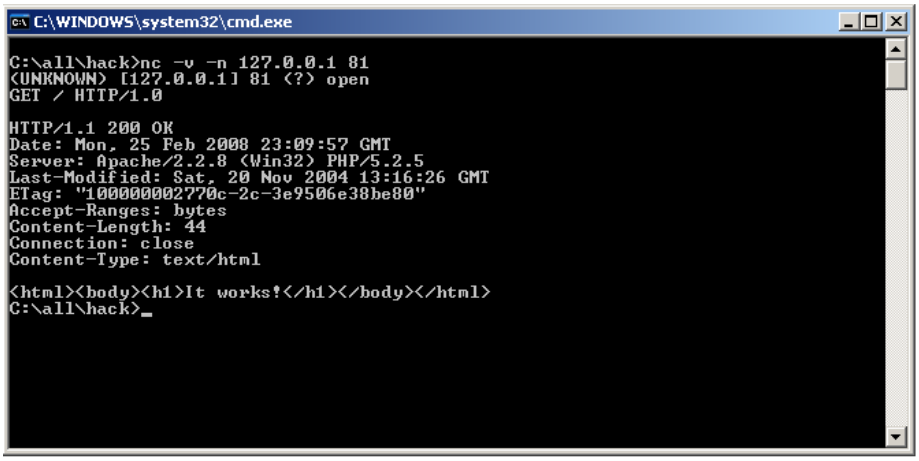
Příkazem „?“ zobrazíme nápovědu, z ní zjistíme, že příkazem „o server port“se připojíme na vzdálený server. Také zadáme příkaz „set localecho“ čímž klientovi říkáme aby znaky pouze neodesílal na vzdálený server, ale také ukazoval u nás na obrazovce. Pak zadáme „o 127.0.0.1 81“ čímž se napojíme na místní www server běžící na portu 81. Poté můžeme zadávat příkazy http protokolu. Nejjednodušší možný příkaz: „GET / HTTP/1.0” (dvakrát odentrovat) nám vrátí tento výsledek:



Obrázek 47: HTTP komunikace z příkazového řádku

Netcat

Telnet je nástroj, který je standardně v každé verzi OS Windows, nicméně práce s ním není příliš uživatelsky přívětivá a jeho možnosti jsou poměrně malé. Mnohem lepší nástroj je například netcat. Do příkazové řádky napište „nc –v –n 127.0.0.1 81“.



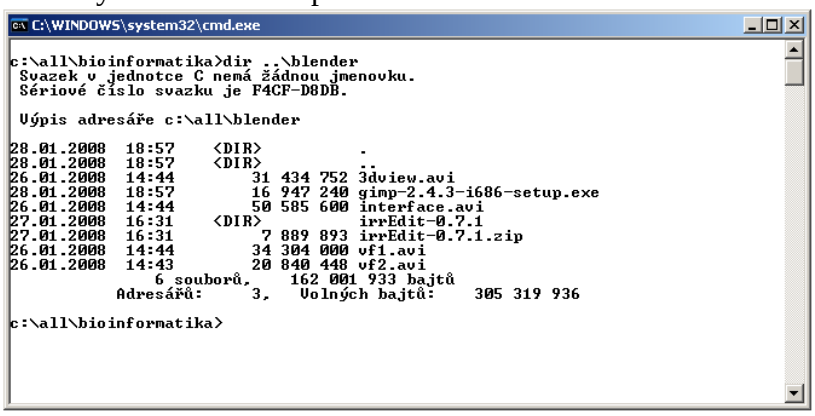
Obrázek 48:http příkaz přes netcat

Úkoly pro cvičení:

1) V připraveném operačním systému ve VMWare prostředí (např. Windows XP) stáhnout z internetu, nainstalovat a nakonfigurovat www server Apache 2.x. Otestovat funkčnost přes prohlížeč a adresu localhost. Poté změnit port na 81, otestovat. Vytvořit adresář chráněný heslem, otestovat jej pomocí nástroje Brutus AET2 (některé antiviry hlásí v tomto nástroji virus i když je stažený z oficiálních stránek autora hoobie.net, jedná se o falešný poplach) – pokusit se odhalit heslo do adresáře hrubou silou.
2) Nainstalovat operační systém Windows 2003 sp2 ve VMWare prostředí, zprovoznit na něm IIS Server a zabezpečit jej podle zásad uvedených v této kapitole.
3) Zabezpečit některý z virtuálních adresářů Basic ověřováním přístupu
4) V připraveném virtuální prostředí s Windows 2000 a nainstalovaným serverem IIS 5.0 (verze zranitelná vůči některým typům útoku directory traversal, laboratoř je nastavená tak, aby pro vás byl tento systém vzdálený, tzn. přístupný pouze jako standardní webový server) se pomocí sekvencí pro procházení nadřazených adresářů pokuste přes zadánímsprávně URL adresy ve webovém prohlížeči spustit soubor c:\windows\system32\cmd.exe. Pokud se to povede, pomocí ve Windows zabudovaného klienta TFTP stáhněte z jiného, připraveného serveru (běží na něm

server protokolu TFTP, např zdarma dostupný tftd32), program netcat (nc.exe). Opět přes URL adresu zadejte příkazy, pomocí kterých netcat nastavíte tak, aby persistentně naslouchal v režimu serveru na vámi určeném portu a přesměrovával svůj vstup a výstup na program cmd.exe. Tím získate vzdálený shell. Ověřte tak, že se z jiného systému připojíte, například opět pomocí netcatu.

Poznámky k úkolu č. 4:
Directory traversal je útok proti webovým serverům nebo webovým aplikacím který využívá špatné kontroly URL (adres www stránek). Klíčem k celému útoku je nedostatečná validace znaku / nebo \ (v závislosti na operačním systému). Pomocí tohoto znaku je totiž možné dostat se ven z adresáře ve kterém se nacházíme do rodičovského adresáře. Toho je možné dosáhnout sekvencí ../ (tečka tečka lomítko). Základní princip je možné vyzkoušet například pomocí příkazové řádky ve Windows a příkazu dir:



Přesto že jsme v adresáři c:\all\bioinformatika vypíše se nám při správném zadání adresář c:\all\blender.

V čisté nezakódované formě (../) je možné tento útok provést naprosto výjimečně protože každá služná webová aplikace tuto sekvenci odfiltruje, nicméně ve spojení s kódováním UTF-8 (unicode) se významně zvyšuje pravděpodobnost že bude útok úspěšný. Kódování unicode využívá k uložení textu 16 bitů (na rozdíl od klasického ASCII, které používá 8 bitů). Jeden znak unicode tedy může nabývat 2^16 hodnot (65536) což umožňuje pomocí jedné znakové sady vyjádřit všechny světové národní abecedy. Nevýhodou je že v takto velké znakové sadě existuje více možností jak zakódovat jeden znak, v případě directory traversal útoku konkrétně znak lomítka / nebo \.

Příklady nebezpečných sekvencí:

- `%2e%2e%2f` se přeloží na `../`
- `%2e%2e/` se přeloží na `../`
- `..%2f` se přeloží na `../`
- `%2e%2e%5c` se přeloží na `..\`

Tyto unicode sekvence se přeloží na / nebo \:

- `%c1%1c`
- `%c0%9v`
- `%c0%af`

Nebezpečí tohoto útoku spočívá ve dvou aspektech – jednak je možné dostat se k obsahu adresářů, ke kterým nemáme mít přístup a jednak pokud je v adresáři, ve kterém se nacházíme povoleno spouštění souborů, může to vést až ke spuštění příkazů na vzdáleném systému. V dnešní době už není útok directory traversal proveditelný přímo na webových serverech, nicméně webové aplikace tuto chybu občas obsahují.

Úkoly pro seminární práce

1) Base64 kódování : naprogramovat v jazyce C (C++, C#) funkci která má vstup ASCII řetězec a výstup Base64 řetězec. Pro pokročilé – naprogramovat v jazyce C (C++, C#) program který tuto funkci využívá ke zkoušení hesel na zaheslovaný adresář s basic autentizací na www serveru.

Pozn. Base64 je kódování které se využívá při http basic autentifikaci. Princip je v tom, že každé 3 byty původního ASCII textu jsou nahrazeny 4 byty, viz. například wikipedia.

Text kontent	**M**								**A**								**n**							
ASCII	77								97								110							
Bit pattern	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
Index	19								22				5				46							
Base64-Encoded	**T**								**W**				**F**				**u**							

2) Ve VMWare nebo VirtualPC nainstalujte Windows Server 2003 (nebo 2008, plně funkční verze jsou zadarmo ke stažení na www.microsoft.com), nainstalujte v něm IIS službu (tzn. role Aplikační server) a certifikační autoritu. Pomocí certifikační autority si vytvořte self-signed certifikát pro webový server, použijte jej pro web v IIS a nastavte v něm HTTPS spojení. Celý postup instalace a nastavení popiště.

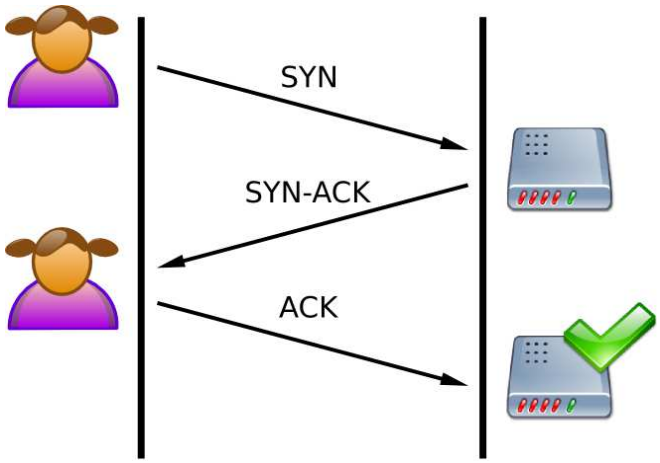
5 Skenování portů

Skenování portů je proces kdy se snažíme zjistit, jaké síťové služby běží na dané IP adrese. Využívají jej útočníci k identifikaci operačních systémů a aplikací na nich běžících. Poté co jsou identifikují síťové služby se snaží zjistit konkrétní aplikaci která je poskytuje a její verzi. Další krokem je prohledání databází známých chyb. Pokud má útočník štěstí, daná verze aplikace má chybu kterou jde zneužít. Pokud ne, může se ještě pokusit aplikaci otestovat pomocí obecných technik. Skenování portů však mohou využít i správci sítě a uživatelé počítačů k tomu aby zjistili jaké porty jsou na jejich systémech otevřeny a tak např. odhalili trojské koně.
Porty jsou čísla, která identifikují síťovou službu na transportní vrstvě. Jsou dva typy: TCP a UDP. Jsou značeny čísly od 0 do 65535. Porty 0-1024 jsou označeny jako privilegované, vyšší porty jako volné.

Nejznámější služby a jejich TCP porty:
20 – FTP data
21 – FTP příkazy
22 – SSH (secure shell)
23 – telnet
25 – SMTP
42 – WINS (Windows Internet Name Service)
53 – Whois
69 – TFTP
80 – http
110 – POP3
119 – NNTP
135 – 139 NetBIOS
143 – IMAP4
194 – IRC
411 – Direct Connect Hub Port
412 – Direkt Connect Client to Client
443 – HTTPS
445 – SMB (Server Message Block) – sdílení souborů a tiskáren ve Windows

Stavy port
Port může být **otevřený**, což znamená, že na něm běží nějaká služba. S takovým portem lze navázat spojení. Opačný stav je, když je port **uzavřený** – neběží na něm žádná služba. Při pokusu připojit se k zavřenému portu je poslán zpět TCP paket s příznaky RST a ACK, v případě UDP portů je zpět poslán ICMP paket typu 3 kód 3 (port nedosažitelný). Poslední možností je že je port **filtrovaný** – to je tehdy když je počítač chráněn firewallem který na daný port nepouští žádný provoz. Ať už je port na počítači za firewallem otevřený nebo zavřený, při pokusu o spojení na filtrovaný port nepřijde žádná odpověď.

5.1.1 Scanovací techniky
Úplné TCP spojení. Útočník se připojí na cílový TCP port a projde celým procesem třícestného podání ruky tak je to vyžadováno RFC (postupně jsou zaslány pakety s nastavenými příznaky SYN, SYN/ACK, ACK). Tuto skenovaní techniku cílový systém snadno pozná, protože zůstane zapsána minimálně v logu aplikace a není příliš rychlá.



Obrázek 49: Plné TCP trojcestné podání ruky

TCP SYN sken. Této technice se říká napůl otevřený sken protože úplné TCP spojení není při tomto typu skenu dokončeno. Útočník na cílový port zasílá SYN paket a pokud cílový stroj odpoví kombinací SYN/ACK, dá se usuzovat, že na portu někdo naslouchá. Pokud vrátí pakety s příznakem RST/ACK, port je pravděpodobně zavřený. Útočník poté zašle RST/ACK, takže spojení místo úplného navázání zruší. Technika je méně nápadná.

TCP FIN sken. Při tomto typu skenu útočník začíná FIN paketem a pokud je cílový port zavřený, měl by operační systém odpovědět paketem RST. Implementace v OS Windows se ale neřídí příslušným RFC dokumentem, takže tato technika funguje pouze na linuxové systémy.

Xmas Tree TCP sken. Útočník začíná kombinací příznaků FIN, URG a PUSH. Pokud je cílový port zavřený, má protistrana odpovědět RST. Název techniky je odvozen od toho, že jsou příznaky TCP paketu „rozsvíceny jako vánoční stromek“.

Null TCP sken. Útočník se představí paketem, který je zcela bez příznaků. Pokud je cílový port zavřený, má protistrana odpovědět paketem s RST příznakem.

TCP ACK sken. Používá se pro analýzu firewallu. Některé jednoduché paketové filtry totiž propouští pouze navázaná spojení (tedy spojení s ACK bitem), takže je lze pomocí paketu s ACK bitem odlišit od stavových firewallů, které provoz na síti sledují podrobněji.

TCP window sken. Tato technika využívá toho že některé systémy za jistých okolností nastavují nestandardní velikost TCP okna. S její pomocí může útočník najít otevřené porty a také porty, které jsou filtrovány.

TCP RPC sken. Používá se pouze na Unixy a slouží k nalezení portů používaných pro RPC (Remote Procedure Call, vzdálené volání procedur). Kromě portů lze touto technikou zjistit také jméno programu, který na příslušném portu naslouchá a jeho verzi.

UDP sken. Útočník na cílový port pošle UDP paket a pokud se mu vrátí ICMP paket typu port unreachable, port je pravděpodobně zavřený. Pokud se IMCP paket o nedosažitelnosti portu nevrátí, dá se usuzovat, že port je otevřený. UDP je ale bezstavový protokol, takže přesnost UDP skenu závisí na mnoha faktorech jako je zatížení sítě nebo nastavení firewallu. Výsledky UDP skenování přes Internet tedy nemusí být přesné.

5.1.2 Nástroje pro skenování portů

netcat

Netcat je univerzální nástroj, často nazývaný švýcarský nůž hackera. Nikoliv překvapivě tedy zvládá i TCP a UDP sken. Parametr –z znamená že má netcat ihned po navázání spojení komunikaci ukončit, parametr –v že má vypisovat podrobné informace (-vv velmi podrobné) –w nastavuje časový limit pro čekání na odpověď, parametr –u zapíná UDP sken (standardní je TCP sken).

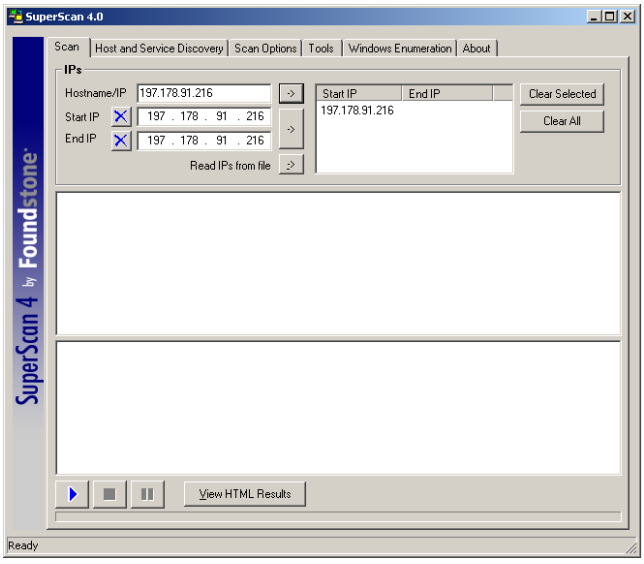
```
nc –v –z –w2 195.178.89.223 1-300
nc –u –v –z –w2 195.178.89.223 1-300
```

amap

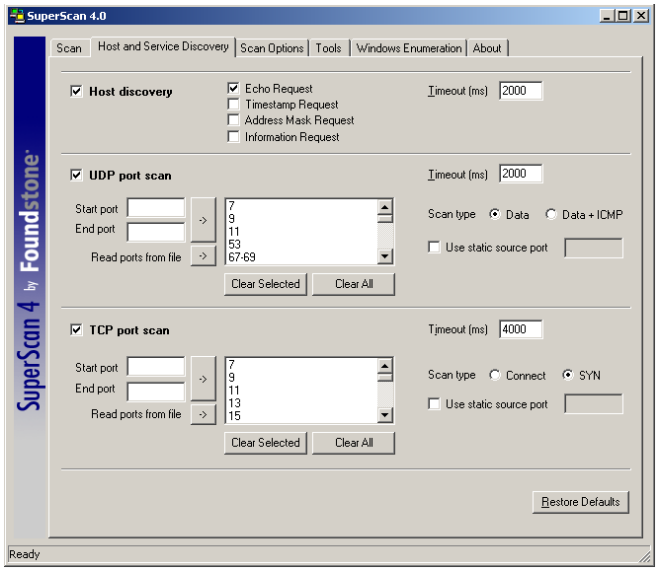
Zajímavý nástroj, který se při identifikaci služby běžící na otevřeném portu nespokojí pouze s tím že je port otevřený ale také zkoumá signaturu služby zde běžící. Díky tomu dokáže správně identifikovat i služby které správce počítače záměrně umístil na jiné než standardní čísla portů. Tento nástroj je zdarma, k dispozici je jak pro Linux tak pro Windows.

SuperScan

SuperScan4 je od známé firmy Foundstone. Je jeden z nejrychlejších, nejspolehlivějších a nejlépe vybavených port skenerů pro Windows. Ovládá TCP i UDP a je zdarma. K hledání živých systémů umí SuperScan využívat čtyři různé ICMP techniky, od standardních ICMP echo paketů až po méně používané ICMP pakety typu timestamp, address mask a information request. Každý z typů ICMP paketů může do výsledného seznamu IP adres přidat další systémy, které lze následovně oskenovat pomocí UDP (režimy Data, Data+ICMP) nebo TCP (full connect sken, SYN sken). K dispozici je také záložka Tools která obsahuje celou řadu užitečných nástrojů – překlad IP adres, ping, ICMP traceroute, zone transfer, hromadný překlad IP adres, http head, http get, http get, vyhledávání v databázích Whois, CRSNIC Whois, ARIN Whois, RIPE Whois a APNIC Whois.



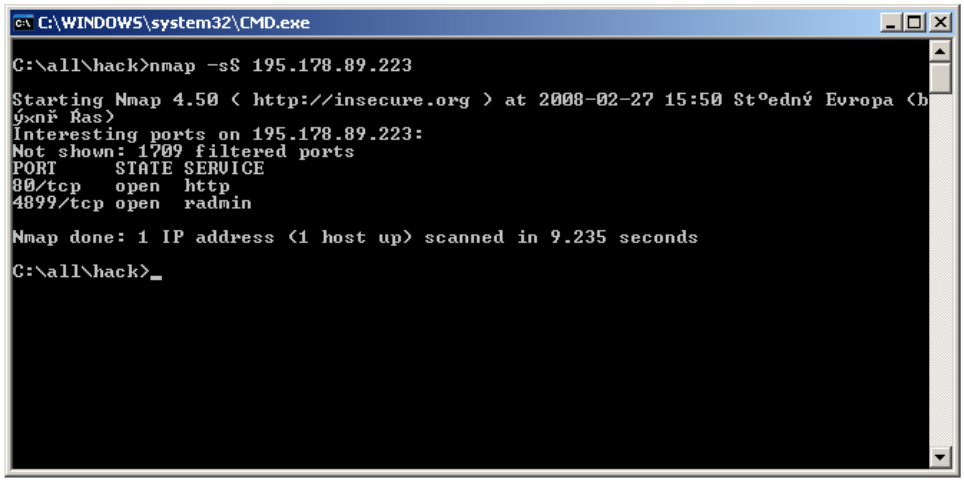
Obrázek 50: Superscan



Obrázek 51: Nastavení scanu v Superscan

nmap

Zcela nejlepší port scanner. Dostupný zadarmo, ve verzi pro Linux i Windows. Kromě základních TCP a UDP skenů zvládá všechny z pokročilejších technik. Ukázka stealth skenu jednoho počítače:

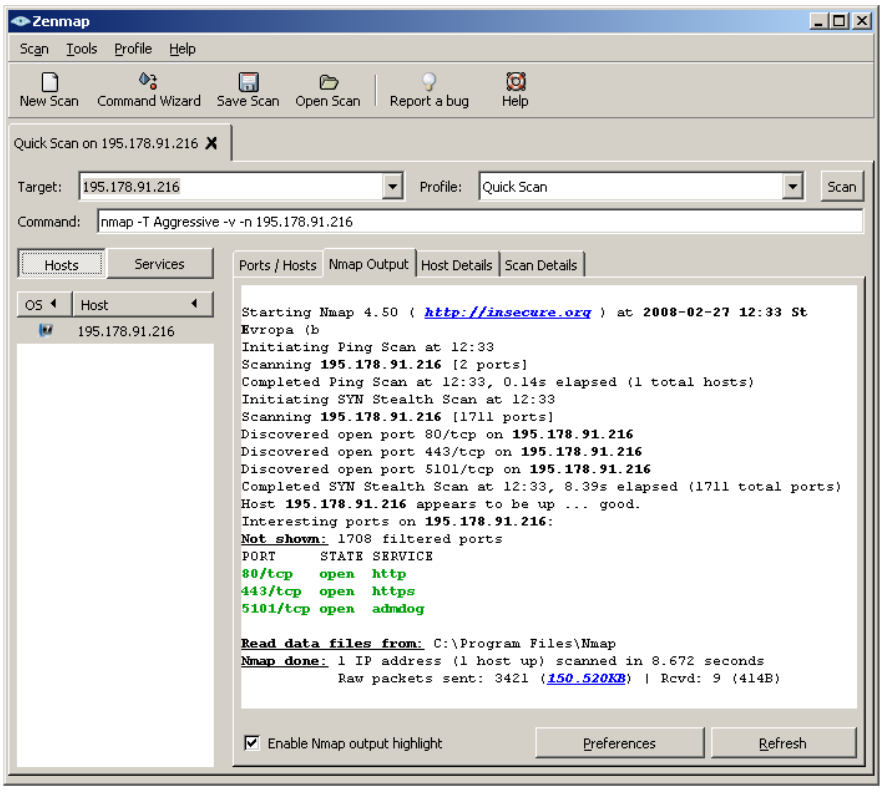


Obrázek 52: NMap

Pokud chceme nmapu zadat větší množství cílů, musíme to udělat pomocí CIDR notace.

zenmap

Zenmap je grafická nadstavba pro nmap. Nabízí naprosto stejné možnosti, pouze je pohodlnější na použití pro uživatele zvyklé na grafické rozhraní a pro ty, kteří nepotřebují výstup programu zpracovávat ve skriptech.



Obrázek 53: Zenmap

Společné použití nmap a amap

Programy nmap a amap lze s výhodou zkombinovat pro některé speciální úkoly. Jako první si představme situaci kdy si uživatelé skrývají nepovolené služby (instant messengery, sdílení souborů, programy pro vzdálený přístup) za povolené porty. Nejprve použijeme nmap k zjištění všech otevřených portů, seznam si uložíme do souboru a ten použijeme jako vstup pro program amap. Ten projde nalezené porty a určí jaké služby na portech běží:

```
nmap –oG nmap-out.txt 195.178.113.145
amap –i nmap-out.txt –o amap-out.txt
```

Dalším situace ve které by se nám mohla hodit kombinace těchto dvou programů je tehdy když chceme určit zda a jaké konkrétní webové servery běží v daném segmentu sítě. Nmap nám rychle nalezne otevřené porty 80 a s pomocí amapu zjistíme co na nich konkrétně běží:

```
nmap –oG nmap-out.txt –p 80 195.178.113.0-255
amap –i nmap-out.txt –o amap-out.txt
```

Výstupy z druhého příkladu by mohly vypadat například takto:

nmap-out.txt

```
# Nmap 4.65 scan initiated Fri Jun 27 10:28:16 2008 as: nmap.exe -oG nmap-out.txt -p 80 195.178.113.1-255
Host: 195.178.113.1 ()  Ports: 80/closed/tcp//http///
Host: 195.178.113.2 ()  Ports: 80/filtered/tcp//http///
Host: 195.178.113.4 ()  Ports: 80/filtered/tcp//http///
Host: 195.178.113.6 ()  Ports: 80/open/tcp//http///
Host: 195.178.113.7 (abc.utb.cz)          Ports: 80/open/tcp//http///
Host: 195.178.113.13 ()Ports: 80/open/tcp//http///
..(zkráceno)..
```

amap-out.txt

```
amap v5.2 (www.thc.org/thc-amap) started at 2008-06-27 10:45:13 - MAPPING mode
Protocol on 195.178.113.6:80/tcp matches http
Protocol on 195.178.113.6:80/tcp matches http-apache-2
Protocol on 195.178.113.15:80/tcp matches http
Protocol on 195.178.113.15:80/tcp matches http-apache-2
Protocol on 195.178.113.18:80/tcp matches http
Protocol on 195.178.113.18:80/tcp matches http-apache-2
..(zkráceno)..
```

Pokročilé průzkumné nástroje – Paketto Keiretsu

Paketto Keiretsu je balík velmi užitečných průzkumných nástrojů jehož první verze vyšla v roce 2002. Jeho autorem je Dan Gaminsky. Povedlo se mu využít do té doby málo prozkoumané techniky použití protokolu TCP/IP. Paketto Keiretsu obsahuje tyto nástroje:

- scanrad: extrémně rychlý bezstavový TCP port scanner
- minewt: směrovač a překladač adres, který běží v userspace
- linkcat: obdoba nástroje netcat pro druhou síťovou vrstvu, bere data ze standardního vstupu a vysílá je skrze ethernetovou kartu na síť
- paragrafe: méně nápadná varianta traceroute, schopná obejít i některé firewally
- phentrophy: nástroj pro grafické zpracování dat, vhodný pro analýzu náhodných čísel

Scanrad

Programy jako nmap, amap, superscan patří mezi „klasickou“ základní výbavu hackerů již řadu let. Tyto nástroje pracují zhruba takto:

- odeslání průzkumného paketu
- uložení informace o tom jaký paket byl na jakou adresu poslán
- čekání na to až se vrátí odpověď

Tento modus operandi má jednu nevýhodu: pokud scanujeme opravdu velké množství počítačů a jejich portů musíme otevřít velké množství síťových spojení a také spotřebujeme velké množství operační paměti protože je nutné po jistou dobu uchovávat informace pro každý pokus o spojení – TCP je totiž stavový protokol. V pamětí musí po celou dobu pokusu o spojení zůstat uložena informace o tom jaký paket byl na jakou adresu poslán. Důsledkem je poměrně velká časová náročnost rozsáhlých skenování.

Scanner scanrad pracuje jinak. Neuchovává žádné informace o tom na jakou adresu poslal průzkumný paket, prostě naplno posílá pakety. Jak ale pozná které z příchozích paketů na jeho síťové rozhraní jsou odpovědí na jeho scan? Při běžném provozu může na takové rozhraní přicházet tisíce jiných paketů od všech možných programů. Dan Gaminsky přišel s řešením nazývaným „inverse SYN cookies“ (obrácené SYN cookies). V principu jde o to že informace o cílové adrese, cílovém portu, zdrojové adrese a zdrojovém portu + tajný klíč jsou zakódovány do ISN (initial sequence number) hlavičky TCP protokolu. Všechny tyto informace by se tam samozřejmě nevešly, proto jsou prohnány hashovaní SHA-1 funkcí která ze 160 bitů udělá 32 bitový otisk. Pokud paket dojde na cílový systém, ten k ISN připočte jedničku a paket pošle nazpátek (s hlavičkami SYN/ACK nebo RST/ACK). Každý došlý TCP paket od systému který odpověděl si tak nese informaci to tom že patří mezi pakety které byly použity ke scanování. Přijímacímu procesu stačí odečíst od AN (acknowledgement number) došlého paketu jedničku, vzít zdrojovou IP adresu+port, cílovou IP adresu+port + tajný klíč, prohnat tyto data SHA-1 funkcí a oba hashe srovnat. Pokud souhlasí, jedná se o paket který byl odeslaný námi (díky tajnému klíči). Díky tomu mechanismu dosahuje scanrad až absurdní rychlosti – síť třídy B (přes 65 tisíc možných strojů) je na dostatečně rychlé lince schopen oskenovat za méně než deset sekund (!!). Mezi jeho další výhody patří to že se skládá ze dvou

samostatných procesů – jeden průzkumné pakety vysílá, druhý naslouchá a čeká na vrácené odpovědi. Protože si ale procesy nemusejí předávat žádnou informaci o tom, které pakety byly odeslány/přijaty je možné každý z procesů spustit na jiném počítači! Tím lze dosáhnout zvýšené úrovně anonymity. Tento nástroj je v současné době dostupný pouze pro unixovské operační systémy.
Parametry:
–b<číslo>: nastavuje propustnost sítě, aby scanrad neodesílal pakety příliš rychle. Omezit lze na daný počet bajtů, kilobajtů, megabajtů, gigabajtů za sekundu. Např. –b1M omezí rychlost skenování na 10Mb/s.
quick: dává se za ip adresu na místo portů. Budou scanovány pouze porty nejběžnějších služeb (80, 443, 445, 53, 20, 21, 22, 23, 25, 135, 139, 8080, 110, 11, 143, 1025, 500, 465, 993, 31337, 79, 8010, 8000, 6667, 2049, 3306). Další možnosti jsou:
squick: super-quick, testuje pouze porty 80, 443, 139, 21, 22, 23
known: čísla vedené u organizace IANA + číslo v souboru nmap-services, celkem 1150 portů
all: všechny porty od nuly včetně do 65 535
- e: scanner čeká i pakety ze zavřených portů, tzn. ty s nastavenými příznaky RST/ACK. Pokud takový paket přijde, dá se předpokládat že mezi námi a cílovým serverem není firewall protože ty obvykle RST/ACK pakety zahazují.
- S: send režim
- L: listen režim

Použití:
Rychlé proskenování jednoho počítače:

```
scanrad –b10M 195.178.113.90:quick
```

Proskenování všech portů jednoho počítače, vysílající počítač je skrytý, přijímající je 203.129.1.13:
Zdrojový počítač:

```
scanrad –t0 –L –s tajny_klic
```

Naslouchající počítač:

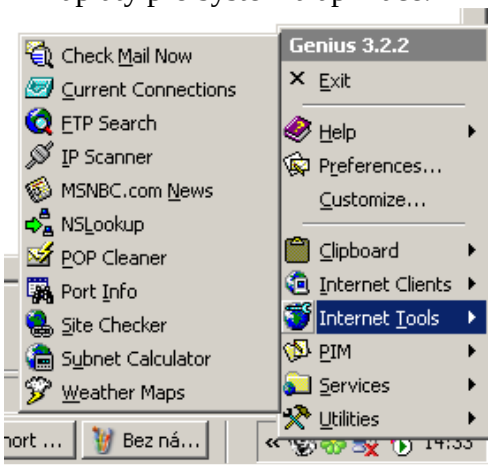
```
scanrad –S –b1m –s tajny_klic –i 203.129.1.13 177.189.223.1:all
```

5.1.3 Obrana proti skenování portů

Možnosti obrany jsou dvě – zapnutý osobní firewall a detekce. V zásadě ale platí že pokud chceme mít veřejně dostupnou síťovou službu, jejímu oscanování zabránit nemůžeme. Hlavním nástrojem pro detekci port skenů jsou síťové IDS (Instrusion Detection Systems) z nichž nejznámější je Snort (dostupný jak pro linux tak pro Windows). Snort je kvalitní a přitom bezplatně šířený nástroj. Na unixových systémech je k dispozici několik dalších utilit – například scanlog. Některé programy lze dokonce nastavit tak, aby na pokus o port sken reagovaly a pomocí firewallu například zakázaly přístup z útočníkovi adresy. Podobná taktika ovšem má i obrácenou stranu mince, protože ji útočník může použít proti nám tím že použije falešné IP adresy a tím zablokuje legitimním počítačům přístup k našim službám. Většina firewallů port sken pozná. Konkrétní firewally se liší schopností odhalit méně nápadné skenovaní techniky, některé odhalit dokáží, jiné ne.

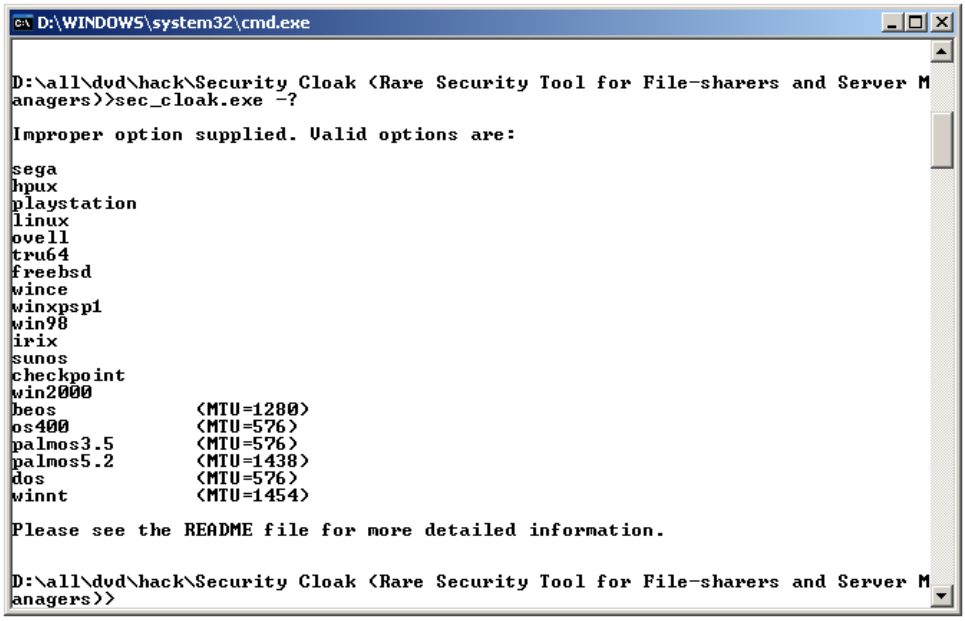
Jednoduché pokusy o skenování portů počítače se systémem Windows dokáže zachytit program Genius od firmy Independent Software (http://www.indiesoft.com). Je to mnohostranný síťový nástroj, který má mimo jiné funkci „Port guardian“ (po instalaci není defaultně zapnutá) která při pokusu o otevření většího počtu portů v krátkém čase upozorní uživatele vyskakovacím oknem.

Zabránit útočníkům a zvědavcům ve skenování je nemožné. Jediné co jde dělat je zajistit jistou míru prevence proti případným následkům – udržovat bezpečnost počítače na co nejvyšší úrovni, tzn. mít zapnuté pouze ty síťové služby, které jsou potřeba, zapnutý firewall, IDS systém a využívat aktuální záplaty pro systém a aplikace.



Obrázek 54: Program Genius

Pokud jde o detekci zjišťování typu a verze operačního systému který běží na našem stroji, máme opět pouze omezené možnosti obrany. Každý operační systém je typický tím jaké porty má otevřené (311/TCP bývá obvykle pouze Mac OS X Server admin, 445/TCP Microsoft Active Directory, 445/UDP Windows SMB sdílení souborů, 22/TCP SSH – může běžet i na jiných ale obvykle Linux/Unix) a jaké konkrétní služby na otevřených portech běží (IIS na linuxu nenajdete). Dále je možno operační systém identifikovat podle toho jakým způsobem nastavuje hlavičky a volitelné parametry síťových protokolů jako například TCP a IP. Jedná se o parametry jako je TTL (Time To Live – hodnota určující po kolika skocích mezi routery je na cestě internetem paket zahozen), time stamp, pmtu (maximum transmission unit – velikost největšího paketu který projde celou cestu v síti bez toho že by byl fragmentován), urg (bitová vlajka oznamující zda jsou přítomny „urgent“ volitelné hlavičky), Windows (velikost TCP okna), sack (selective acknowledgement).
Jednou z možností jak zamaskovat svůj operační systém je změnit standardní čísla portů na kterých běží naše služby za jiné. Tím ale můžeme jednat způsobit nekompatibilitu některých klientů a hlavně tím nezabráníme tomu aby se důsledný útočník o službě stejně nakonec dozvěděl. Tento princip se nazývá „security by obscurity" a není na něm nic špatného pokud je použit jako doplňující linie obrany. Bylo by ale kritické spoléhat pouze na něj. Další možností kterou máme je změnit výchozí parametry síťových protokolů tak aby náš systém podle těchto charakteristik vypadal jako něco jiného. Na OS Windows nám k tomu poslouží program Security Cloak. Je to jednoduchý program pracující z příkazové řádky, který změnou hodnot v registru systému Windows vyvolá požadované změny. Umožňuje nám tvářit se jako jeden ze dvaceti operačních systémů (z toho některé velmi exotické jako například OS konzole Sega Dreamcast).



Obrázek 55: Security Cloak

Nahlédnutím do zdrojových kódů můžeme zjistit, jaké parametry jsou pro jakou volbu nastaveny:

```
……
else if(strcmp(string,"linux")==0){
            ttl=64;
            stamp=0;
            pmtu=0;
            urg=0;
            window=16384;
            sack=0;
            //setmtu(1500);
        } else if(strcmp(string,"novell")==0){
            ttl=598;
            stamp=0;
            pmtu=1;
            urg=0;
            window=16384;
            sack=0;
            setmtu(1500);
        } else if
…..
```

Stránka autora ani stránka samotného nástroje na ní (http://www.craigheffner.com/security/) není v době psaní tohoto textu bohužel k dispozici, nicméně nástroj je možné nalézt například na torrentech.

Aktivní obrana proti skenování portů

Existují dvě možnosti jak se aktivně bránit skenování portů:

- úprava jádra operačního systému tak aby neposílal RST pakety (prakticky proveditelné pouze u OS linux) – výsledkem je nefunkčnost velkého množství scanovacích technik, hlavně těch skrytých
- předstírání otevření velkého TCP množství portů (kombinací tcpdump pro odchytávání skenu a hping2 pro generování falešných odpovědí) – výsledkem je zahlcení útočníka informacemi protože se mu jeví že jsou otevřené všechny TCP porty a služby tudíž musí hledat jinak

Úkoly pro cvičení:
K těmto cvičením je nutné připravit virtuální stroje různých operačních systémů a zpřístupnit je po síti (ať už skutečné nebo pouze virtuální).

1) Zjistěte živé počítače ve své podsíti
2) Zjistěte, jaké operační systémy na nich běží
3) Zjistěte, jaké služby na nich běží a pokuste se zjistit jejich verze. Zkontrolujte je s databází chyb zda jsou některé z nich zranitelné (databáze Bugtraq na securityfocus.com)
4) Ručně (spojte se pomocí netcat nebo telnet) zjistěte, na kolika z nich běží webové servery + jejich výrobce a verze
5) Pomocí nástroje nmap oskenujte libovolný systém. Vyzkoušejte všechny typy skenů které nmap nabízí (full, x-mas, ..). Jednotlivé skeny zachyťte pomocí programu Wireshark (nebo jiného snifferu) a zkontrolujte/popište jejich princip fungování (jaké pakety jsou posílány) na základě analýzy zachycených dat.

Úkoly pro seminární práce:
1) Vypracujte seminární práci na téma možnosti programu nmap. Vycházejte z nápovědy k aktuální verzi programu. Popište jednotlivé typy scanů, které s programem lze provést, uveďte příklady.

6 Google Hacking

Vyhledávač Google se v posledních pěti letech stal neodmyslitelným pomocníkem všech uživatelů Internetu. Je přirozené, že jej využívají i hackeři a bezpečnostní komunita. Kromě vyhledávání návodů, nástrojů, dokumentací, zdrojových kódů a dalšího lze Google zneužít také přímo k hledání zranitelných cílů, emailových adres a k mapování sítě cizích organizací.
Společnost Google byla založena v roce 1998 dvěma Stanfordskými studenty Larry Pagem a Sergeyem Brinnem. Původně studentský projekt experimentálního vyhledávače se vyvinul v jednu z nejúspěšnějších a největších společností světa. Hlavní aktivitou společnosti a zároveň zdrojem příjmů je vyhledávací engine, který je již od svého vzniku právem považovaný za nejlepší vyhledávač na Internetu.
Vyhledávání v googlu je proces jehož cílem je najít informace. Proces začíná základním hledáním které poté modifikujeme tak dlouho dokud nezískáme jen ty stránky které obsahují relevantní informace. Je nutné provést tzv. redukci vyhledávání.

Základní pravidla pro vyhledávání v Googlu:

- nejjednodušší dotaz se skládá z jediného slova nebo z kombinace více slov
- pokud hledáme přesné slovní spojení uzavřeme je do uvozovek „ „. Google pak toto spojení hledá jako celou frázi.
- lze použít booleovské operátory AND, OR, NOT. Operátor AND je redundantní protože google standardně zahrnuje do vyhledávání všechny termíny. Operátor NOT lze použít také tak že před daný termín který **nechceme mít ve výsledcích vyhledávání dáme znaménko mínus** (-), bez mezery (hacker –golf). Tento operátor je jeden z nejdůležitějších protože nám umožňuje postupně zmenšovat množinu výsledků o nerelevantní stránky.
- Google nerespektuje prioritu operátorů – zpracovává je jako větu kterou čte zleva doprava. Pokud chceme dotaz strukturovat musíme použít závorky.
- v dotazech se nerozlišuje velikost písmen. Jedinou výjimkou je slovo or. Chcete-li jej použít jako booleovský operátor musíte jej napsat velkými písmeny – OR
- zástupný znak v Googlu je hvězdička (*), která reprezentuje jedno slovo
- Google automaticky používá zkrácený zápis slov
- Google ignoruje velmi běžná slova, znaky a jediné číslice (tzv. stop words). To že jsou některá slova ignorována je oznámeno na stránce s výsledky. Mezi tyto slova patří například who, where, what, the, a, an. Donutit google aby hledal i běžná slova lze tím že dáme celé slovní spojení ve kterém se vyskytují do uvozovek. Tím bude hledání zpracováno jako celá fráze. Dalším způsobem je použít znak plus: +and, +a. Pokud mezi znaky „+“ a „a“ vložíme mezeru (+ a), budou nalezeny všechny stránky na kterých se vyskytuje samostatné písmeno a. Tím lze poměrně dobře získat přehled o tom kolik stránek google indexuje. V době psaní tohoto textu (červen 2008) to bylo téměř 19 miliard stránek.
- Google limituje hledání na 10 termínů. Do limitu jsou zahrnuty vyhledávané termíny i pokročilé operátory. Existuje způsob jak do dotazu dostat více než 10 termínů – často ignorované slova nahradit zástupným znakem *, který Google nezapočítává jako vyhledávaný termín.

Pokročilé operátory googlu:

- intitle, allintitle
- inurl, allinurl
- filetype

- allintext
- site
- link
- inanchor
- daterange
- cache
- info
- related
- phonebook, rphonebook, bphonebook
- author
- group
- msgid
- insubject
- stocks
- define

Základní syntax pokročilých operátorů má v googlu tvar operátor:hledaný_termín. Mezi operátorem, dvojtečkou a hledaným termínem nesmějí být žádné mezery. V jednom dotazu lze kombinovat více pokročilých operátorů, které se nicméně kombinovat nedají. Operátory začínající na all (allintitle, allinurl, allintext) se obvykle používají jen jednou a nelze je kombinovat s ostatními operátory.

intitle
Vyhledává pouze v titulku stránky (HTML tag <title>). Jako dotaz se bere pouze to co je bezprostředně za dvojtečkou operátoru (slovo nebo fáze v uvozovkách). Další slova nebo fráze z dotazu jsou hledány už v textu. Naproti tomu varianta allintitle vyhledává všechna slova z dotazu pouze v titulku.

allintext
Dotaz bude vyhledáván pouze v textu (jsou tedy vynechány titulek, URL a odkazy).

inurl
Parametr bude vyhledáván pouze v URL (Universal Resource Locator) stránky – v adrese. Pro vyhledávání výhradně v URL je operátor allinrul.

site
Zúží vyhledávání na weby které mu specifikujeme. Např. site:.utb.cz bude vyhledávat pouze na serveru utb.cz ale ve všech jeho doménách (fai.utb.cz, ft.utb.cz, fame.utb.cz,..) kdežto site:fai.utb.cz zúží vyhledávání pouze na danou subdoménu.

filetype
Zúží vyhledávání pouze na soubory daného typu. Celkem existuje asi 8000 různých typů souborů (www.filetext.org). Nejčastější soubory ve kterých lze vyhledávat jsou pdf, ps (adobe postskript), mw (MacWrite), xls (MS Excel), doc, (MS Word), rtf (Rich Text Format), swf (Shockwave Flash), txt. Google většinu z nich dokáže indexovat, tzn. vyhledávat informace uvnitř těchto souborů. Kromě toho také soubory které dokáže indexovat dokáže většinou i převést na formát HTML nebo txt tak aby byl přímo čitelný v internetovém prohlížeči, tzn. k prohlížení například nalezených xls souborů nepotřebujeme mít nainstalovaný MS Excel. Tato funkce je dostupná tehdy když je u výsledku odkaz „View as HTML“.

link
Vyhledává stránky které mají odkazy na jiné stránky (námi zadané). Operátor link nevyžaduje text ale URL nebo název serveru.

inanchor
Vyhledává nikoliv v samotném odkazu jako link ale v textu odkazu (podtržený text viditelný prohlížeči).

cache
Zobrazí archivovanou verzi stránky. V syntaxi je nutné použít celý název server: cache:www.google.cz

numrange
Operátor numrange vyžaduje dva parametry – horní a dolní mez. Oddělují se pomlčkou. Hledá čísla v daném rozsahu. Existují dvě zkrácené verze tohoto operátoru – místo numrange:1-1000 stačí do dotazu zadat jen obě čísla oddělená dvěma tečkami (1..1000). Kromě toho se dá využít operátor ext, ve tvaru ext:1-1000.

daterange
S tímto operátorem lze z vyhledávání odfiltrovat stránky spadající do určitého datumu. Operátor nicméně není podporovaný oficiálně a je nutné do něj datum zadávat jako juliánské datum (=počet dní které uplynuly od 1. ledna 4713 p.n.l.). Pro běžné vyhledávání je lepší použít GUI volbu „Pokročilé vyhledávání“.

info
Info zobrazí souhrnné informace o webu a odkazy na jiná hledání googlu týkající se webu. Parametrem je URL.

related
Related zobrazí ty weby u kterých google určil že jsou podobné zadanému webu. Parametrem je URL.

Vyjmenované operátory jsou hlavní operátory použitelné na vyhledávání www stránek. Ostatní operátory slouží k vyhledávání v Google Groups, telefonních seznamech, akciích apod.
Nové techniky prohledávání jsou k dispozici k otestování na adrese http://labs.google.com – Experimental Search. Ty, které se osvědčí v tomto veřejném betatestu jsou potom nasazeny do hlavního vyhledávače.

Archiv Googlu
Archiv googlu je služba která zpřístupňuje staré verze stránek. Z hlediska počítačové bezpečnosti má význam v tom že zvědavcům nabízí možnost prozkoumat cílový web bez toho že by na něj zaslali jediný paket, čímž dokáže zakrýt jejich aktivity a poskytnout jistou míru anonymity (do té míry že archivovány jsou pouze textové informace, pokud si prohlížeč vyžádá i obrázky, jsou staženy z originálního serveru což už stopy zanechá). V archivu jsou uchovány téměř všechny dokumenty které googlebot navštívil.
Vyhledávání spustitelných souborů
V roce 2006 bylo webovými uživateli zjištěno že Google indexuje spustitelné PE soubory (.exe a .dll a další) – např. dotazem "Signature: 00004550" šlo najít statisíce takových

souborů. Kromě toho že je byl schopen je najít také dešifroval jejich hlavičky a umožňoval v nich vyhledávat. Tato možnost vzbudila poměrně velkou odezvu v bezpečnostní komunitě kvůli faktu že takto nalezené soubory šlo v prohlížeči stáhnout a spustit a tím došlo k ohrožení uživatelům možným malwarem. V době psaní tohoto materiálu už Google možnost vyhledávat spustitelné soubory nenabízí, dotazy které dříve vedly k nalezení exe souborů už nyní poskytnou pouze stránky pojednávací o tomto fenoménu. Vyhledávač společnosti Microsoft Live.com však ano. Stačí na Live.com zadat například následující dotaz: "Time Date Stamp:" "Machine: Intel 386" a jsou nalezeny desetitisíce výsledků.
http://www.quietearth.us/articles/2006/11/14/How-Google-and-Live-Binary-Search-works
http://googlesystem.blogspot.com/2006/06/some-google-results-are-exe-files.html
http://securitylabs.websense.com/content/Alerts/1152.aspx

6.1.1 Pasivní průzkum cílové organizace

Protože Google udržuje archiv všech stránek které jeho googlebot navštíví a dává jej k dispozici přes funkci „Cache“, je možné využít jej k prozkoumání webových stránek bez toho abych dané stránky vůbec museli navštívit a zanechat tedy stopu v jejich logu. Kromě toho lze například prozkoumat usenetové diskuse zda se v nich pracovníci dané firmy nesvěřili s důvěrnými informacemi ve snaze nalézt pomoc při řešení technických problémů (např. jaký software a v jaké konfiguraci je nasazen apod). Lze také zjistit veřejně dostupné subdomény dané firmy – často se na nich mohou nacházet „zapomenuté“ servery jejichž služby už organizace nevyužívá ale nevyřadila je z provozu, nespravuje je a je u nich tedy vyšší riziko výskytu nezáplatovaných chyb v konfiguraci nebo softwaru.

6.1.2 Deset základních bezpečnostních vyhledávání

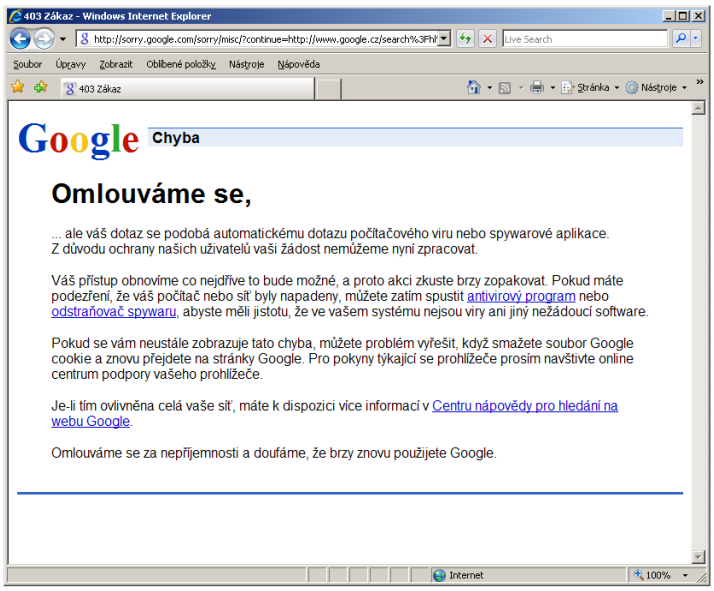
Předem je potřeba zdůraznit se Google „nejprofláknutější“ dotazy na zranitelné webové servery a aplikace filtruje a nezobrazí takže žádné „zaručené“ dotazy neexistují, každý z takových dotazů funguje pouze omezenou dobu. Nicméně princip jak najít nové zranitelné webové aplikace zůstává stejný – je potřeba znát jak zranitelná verze vypadá, začít širokým obecným dotazem a postupně výsledky ořezat tak až najdeme to co potřebujeme. Stejné také zůstávají užitečné operátory a styl myšlení při podobných dotazech a to pomocí následujících příkladů předat jde.

1. site

Operátor site je pro útočníky nedocenitelný ve fázi sběru informací. Jeho použití je především jako základní dotaz pro další vybrušování, při použití bez dalších omezení vrací většinou příliš velký počet výsledků (např. *site:microsoft.com* přes 60 milionů stránek). Velmi užitečná je jeho kombinace s operátorem „-„ když chceme zjistit subdomény nějakého serveru: dotazem „*site:microsoft.com -site:www.microsoft.com*“ lze během pár sekund zjistit že doména microsoft.com má subdomény msdn, technet, crawlmsdn.com, support, uddi, partner, advertising, licencing, grv, reserch, vista.gallery, preview, members, partnerconnect, learning, flexcomp, pinpoint, forums, sba, solutionfinder, cuai97, connect, isys, expression, office, aer, epp, activex, hup, privacy, bac, eagreements, msevents, microsoftclub, oca, referencesource, activate, adcenter, wpf, winqual, points, sc, sharepoint, premiere, building, profile, billing, input, uncertainity99, expertzone, murl, msops, connectbeta, schemas, cuai-96, wweventstest, udlab, windowsupdate, messsaging, eopen, das, netmeeting, mcp, mbs,…

Jak je vidět subdomén může být u velké organizace desítky až stovky. Nelze samozřejmě čekat že u tak exponované firmy jako je Microsoft nalezneme (ať už jakkoliv složitým) dotazem v Googlu zranitelnou stránku, nicméně i zde je vidět spousta výsledků které jistou možnost zneužitelných bezpečnostních rizik slibují – subdomény jako cuai-96.microsoft.com,

cuai97.microsoft.com, uncertainity99.microsoft.com, connectbeta.microsoft.com, beta.microsoft.com nabízejí staré verze nebo betaverze internetových stránek které by u firmy která není tak ostražitá jako microsoft klidně mohly nabízet možnost zneužití například pomocí XSS útoku. Dotaz na subdomény můžeme vybrušovat tím že odečítáme další a další nalezené subdomény a dostáváme se tak postupně k čím dál více obskurnějším výsledkům a tím i potenciálně zanedbanějším stránkám Pokud ale budeme ve svém čmuchání příliš zvědaví (například dotazem „*site:microsoft.com -site:www.microsoft.com -site:partner.microsoft.com -site:research.microsoft.com -site:advertising.microsoft.com -site:support.microsoft.com -site:technet.microsoft.com -site:msdn.microsoft.com -site:search.microsoft.com -site:preview.microsoft.com -site:grv.microsoft.com -site:partnerconnect.microsoft.com -site:uddi.microsoft.com -site:licensing.microsoft.com -site:forums.microsoft.com -site:crawlmsdn.microsoft.com -site:learning.microsoft.com -site:vista.gallery.microsoft.com -site:members.microsoft.com -site:flexcomp.microsoft.com -site:office.microsoft.com -site:pinpoint.microsoft.com -site:sba.microsoft.com -site:solutionfinder.microsoft.com -site:connect.microsoft.com -site:cuai97.microsoft.com -site:isys.microsoft.com -site:expression.microsoft.com -site:aer.microsoft.com -site:oem.microsoft.com*“) můžeme se dočkat následující odstavující obrazovky:



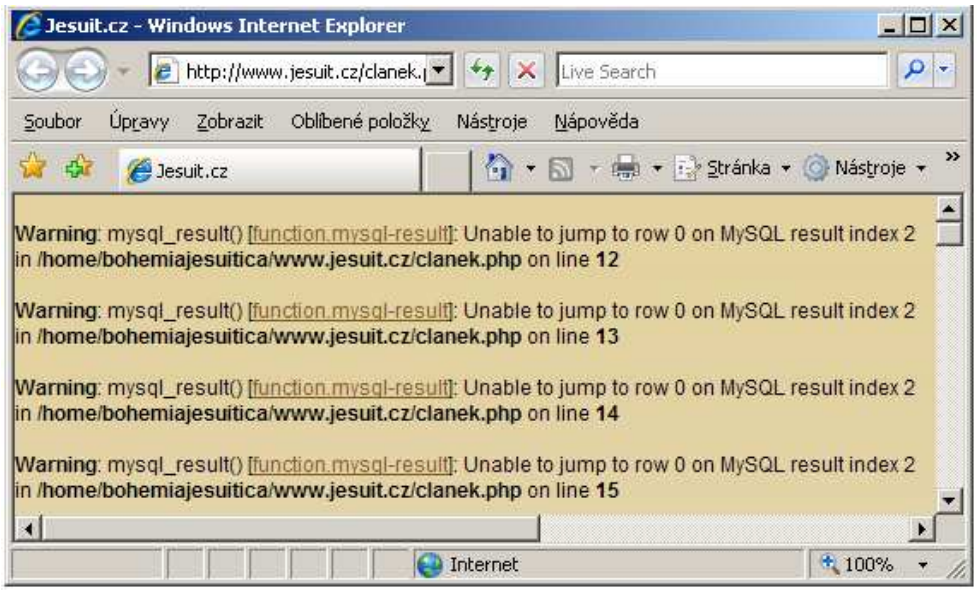
Obrázek 56: Chybová stránka vyhledávání Google

2. intitle:index.of

Univerzální dotaz pro hledání výpisů adresářů. Většinou takto najdeme pouze apache servery, ale těch je na Internetu stejně přibližně 50%. Tečka v tomto dotazu je zástupný znak (za mezeru, jinak bysme museli dát dotaz do úvozovek).

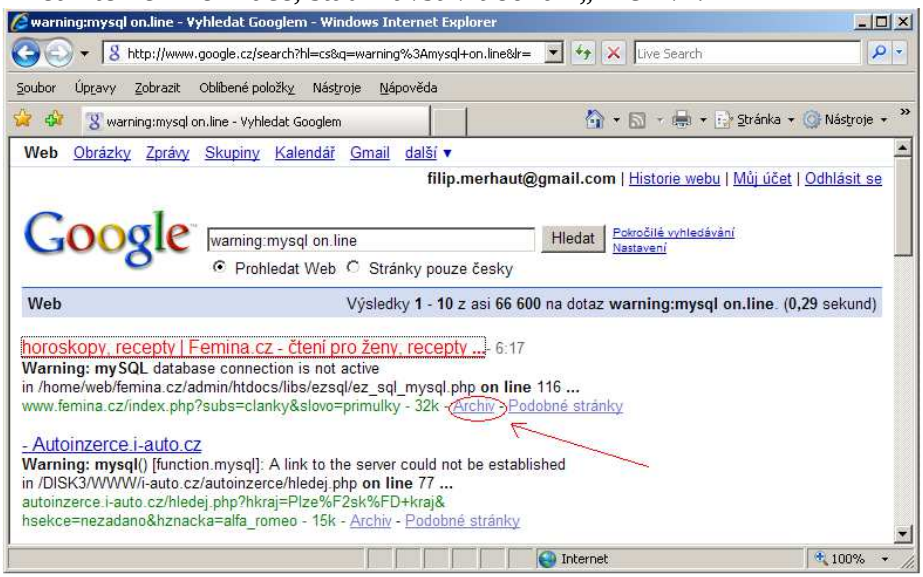
3. error | warning

Chybové informace dokáží o cíli odhalit velké množství informací (typ softwaru, operační systém a jeho verzi, použité moduly softwaru, konfigurační detaily aplikace, v extrémním případě až hesla). Některé chyby nicméně neobsahují přímo slovo error. Záleží na konkrétní webové aplikaci, chyby v PHP při vykonávání MySQL funkcí vypadají například takto:



Obrázek 57: Chyba ve webové aplikaci

Podobné stránky můžeme nalézt například dotazem „*warning:mysql on.line*“. Protože Google nemůže mít každou stránku na Internetu zaindexovánu v nejaktuálnější verzi může se stát že nám na takový dotaz najde stránku která je ve své živé verzi už opravená. I takový výsledek se ale útočníkovi může hodit, Google v podstatě použije jako archiv který může uchovat potenciálně zneužitelné informace, stačí navštívit odkaz „Archiv“:



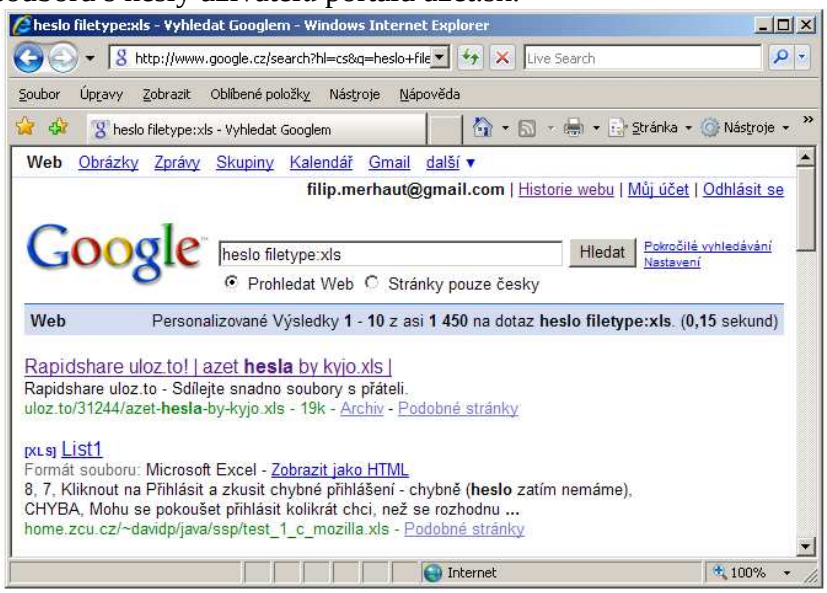
Obrázek 58: Vyhledávání chyb v Google

4. login | logon

Login portály jsou vstupní branou do chráněných webových aplikací. Vyžadují jméno a heslo ale často také prozradí operační systém a software na kterém běží. Občas je také k dispozici odkaz typu „Zapomenuté heslo“ které buď vedou stránku na které musíme zodpovědět záchranou otázku ve stylu Jméno za svobodna (často jde o údaje které se dají opět vygooglovat) nebo poskytnou email, URL či telefonní číslo na podporu – která se může pro dostatečně drzého a pohotového hackera stát obětí sociálního útoku (například nejmenovaný český freemailový portál ještě nedávno řešil zapomenuté heslo tím způsobem že stačilo na podporu uživatelů poslat email z jiné schránky a poprosit o změnu hesla..). Kromě toho se login portály mohou samozřejmě stát terčem útoku hrubou silou.

5. username | userid | employee.ID | "your username is"
Zjištění uživatelského jména není pro hackera vítězstvím samo o sobě, nicméně je to přece jen polovina informací které potřebuje k narušení autentizačního systému a tudíž dobrý začátek.

6. password | passcode | "your password is"
Slovo password" případně česká „heslo" se na internetu vyskytuje velmi často takže takto jednoduše postavené hledání nám poskytne akorát hromadu sena ve které se žádná jehla nakonec vůbec nacházet nemusí. Opět je to ale dobrý základní dotaz pro další obrušování a s dostatkem trpělivosti a tvůrčí fantazie se k reálným výsledkům dopracovat dá. V době psaní tohoto textu například hned první výsledek velmi jednoduchého dotazu „*heslo filetype:xls*" vedl k získání souboru s hesly uživatelů portálu azet.sk:



Obrázek 59: Vyhledávání souborů v Google

7. admin | administrator
Nečekejte, že tento dotaz odhalí na první stránce hesla administrátora. Minimálně ale dokáže odhalit alespoň něco ze struktury webového serveru, stránkách na kterých se provádí správa apod.

8. –ext:html –ext:htm –ext:shtml –ext:asp –ext:php
Pokud vyloučíme standardní přípony souborů pro www stránky, můžeme se dočkat velmi zajímavých výsledků v podobě dokumentů, zdrojových kódů, textových poznámek, konfiguračních souborů a spousty dalšího.

9. inurl:temp | inurl: tmp | inurl:backup | inurl:bak
Není příliš velkou vzácností že si vývojáři webových aplikací z pohodlnosti zálohují starší verze www stránek tím že prostě přidají podobnou příponu. Tím ovšem způsobí že soubor poté půjde přečíst jako text místo toho aby jej server vykonal jako kód. Pokud se takový soubor na serveru zapomene a google jej indexuje, může jít o potenciální katastrofu.

10. intranet | helpdesk
Cílem helpdesku je co nejvíce zjednodušit práci uživatelů – v jistém smyslu jde tedy o protichůdný cíl než má počítačová bezpečnost. Ne vždy jsou systémy helpdesku navrženy tak aby byly neprůstřelné proti drzému sociálnímu útoku.

6.1.3 Stránky zabývající se Google Hackingem

Nejznámější stránka tohoto druhu je http://johny.ihackstuff.com, jejíž provozovatel Johny Long napsal v roce 2005 knihu Google Hacking for Penetration Testers (vyšla i česky pod názvem Google Hacking). Tato stránka několik let udržovala aktuální seznam tzv. Google Hacků neboli také GoogleDork výrazů – dotazů pro vyhledávač Google které umožňovaly nalézt zranitelné systémy, hesla, osobní informace apod.

Vybrané dotazy

Všechny pdf soubory:
filetype:pdf pdf
Sniffer na stránce codeproject:
site:codeproject.com inurl:Sniffer
Seznam operátorů pro google.com
site:google.com inurl:operators
Proxy servery:
"this proxy is working fine!" visit
Všechny subdomény utb.cz:
site:utb.cz -site:www.utb.cz

Adresáře ve kterých jsou ke stažení knihy v nejrozšířenějších elektronických formátech například na téma hacking:
-inurl:(htm|html|php) intitle:"index of" +"last modified" +"parent directory" +description +size +(.txt|.lit|.doc|.rtf|.zip|.rar|.pdf|.chm) "hacking"

Obrázek 60: Příklad vyhledávání

Bugtraq exploitů na serveru insecure.org které obsahují i GoogleDork výraz pro nalezení živého zranitelného systému a jsou z roku 2007:
googledork site:seclists.org inurl:2007
Obdobný výraz pro server securityfocus.com:
2007 googledork site:securityfocus.com

Log soubory od serverů s přístupem placeným přes ccbill. Tyto soubory obsahují přihlašovací jméno pro danou doménu v otevřeném textu + heslo chráněné DES.
inurl:ccbill filetype:log

Dotaz se kterým se dříve (než je Google zablokoval) daly nalézt přímo čísla kreditních karet. Jakou odpověď si myslíte dostaneme na tento dotaz dnes?
visa 4356000000000000..4356000000000000

Úkoly pro cvičení:

1) Vyzkoušejte, kolik informací o Vás mohou najít zvědavci pomocí Googlu
2) Nalezněte pomocí googlu co nejvíce subdomén vybraného velkého serveru (například Microsoft.com, Google.com)
3) Zkuste donutit Google k odmítnutí provedení Vaše dotazu a zobrazení stránky „403 Omlouváme se, ale váš dotaz se podobá automatickému dotazu počítačového viru nebo spywarové aplikace“. Jaký dotaz jste k tomu použili?
4) Nalezněte server IIS 4.0, pomocí nástroje nc.exe (NetCat) zkontrolujte, zda se opravdu jedná o tuto verzi serveru
5) Na serveru Securityfocus.com si v sekci Vulnerabilities vyberte kteroukoliv chybu u softwaru s webovým rozhraním (webové aplikace, webové servery, webové rozhraní routerů, tiskáren, webových kamer, ..) a s pomocí Googlu na Internetu nalezněte konkrétní adresu s tímto zranitelným softwarem (případně alespoň stejný software ve která je již záplatovaná)
6) Připravit falešnou stránku neexistující webové aplikace, nejlépe úpravou některé existující (tak aby prosté vyhledávání v textu vyhodilo hodně falešných výsledků), nahrát tuto stránku na některý veřejný www server, zaslat ji do Googlebotu a poté co bude indexována v Googlu zadat studentům její offline příklad s úkolem najít na Internetu její živou verzi.
7) Nalezněte (např. pomocí operátoru inurl) a prozkoumejte několik různých extranetů. Zjistěte, zda se jejich IP adresa liší od IP adresy hlavní (subdoména www) webové prezentace dané entity. Pokud ano, co z toho lze vyvodit?

7 Buffer overflow - přetečení zásobníku

Přetečení zásobníku je jednou z nejzávažnějších chyb v počítačové bezpečnosti. Zneužití tohoto typu chyb má velmi dlouhou historii a jak ukazuje poslední vývoj (např. bezpečnostní bulletin Microsoftu MS08-067) zdaleka ještě nekončí a to ani přesto že je jí v posledních pěti letech při vývoji softwaru věnována obrovská pozornost. Nejprve je nutné si ujasnit několik pojmů.

7.1.1 Terminologie

Assembler – jazyk symbolických instrukcí. Nejnižší programovací jazyk, používající přímo instrukce procesoru (jejich symbolické kódy).
Buffer (memory) oveflow - Přetečení (paměti) zásobníku -
Vulnerability – zranitelnost, objevené zranitelné místo v programu.
Proof of koncept – důkaz proveditelnosti. Kód demonstrující existenci zranitelného místa ale pouze pro demonstrativní účely, neprovádí žádné zneužití chyby. Schopným programátorem ale může být rozšířen na plnohodnotný exploit.
Exploit – vytěžení. Kód provádějící zneužití vulnerability. Nejčastěji má podobu kódu v jazyce C, assembler nebo perl. Exploit provede
Shellcode – shelkód, kód v assembleru spouštějící vzdáleně přístupnou příkazovou řádku, obecně jakýkoliv kód který chceme provést při spouštění exploitu.
0-day – den nula. Nově objevené slabé místo, nezveřejněné ve veřejných databázích vulnerabilit. Pokud slabé místo není zveřejněno, většinou o něm neví ani samotný výrobce softwaru, tím pádem nevytvoří opravující záplatu a systému na internetu obsahující software s touto chybou budou bezbranné vůči útočníkům majícím k dispozici fungující exploit. 0-day bývají právě z tohoto důvodu velmi cenné a šíří se pouze v uzavřené komunitě black hat hackerů systémem výměny, případně mohou být zpeněženy na černém trhu.
Fuzzing – metodika testování softwaru při vývoji, snažící se pomocí automatických softwarových nástrojů odhalit potenciální místa přetečení paměti

7.1.2 Základní myšlenka útoku

Přetečení paměti nastane tehdy, když do proměnné vložíme více bytů než má tato proměnná přiděleno paměti od operačního systému. Byty vložené navíc se musí někam uložit. Kam přesně to záleží na operačním systému a programovacím jazyce. V některých případech přepíšeme „pouze“ okolní proměnné, sousedící s naší „přetečenou“. V určitých situacích ale můžeme přepsat i tzv. návratovou adresu funkce. Tato adresa je spolu s ostatními daty uložena v tzv. zásobníku (zásobník je dočasné místo pro ukládání data v softwarových aplikacích). Je použita ve chvíli kdy skončí právě vykonávaná funkce. Na tuto adresu se procesor „vrátí“ protože podle konvence na ní má ležet kód odkud byla současně vykonávaná funkce zavolána. Tento kód byl tedy přerušen voláním funkce a nyní je jeho provádění obnoveno. Pokud návratovou adresu přepíšeme, nevrátí se procesor na původní místo ale na jiné místo v paměti počítače – kam, to záleží jakou hodnotou návratovou adresu přepíšeme. Na novém místě v paměti může být cokoliv – data jejichž interpretace ve formě instrukcí nedává procesoru smysl, chráněné místo paměti kam náš proces nemá přístup, jiná funkce našeho programu, funkce operačního systému.. Schopný útočník dokáže za správných podmínek přepsat návratovou adresu přepsat tak aby ukazovala na instrukce, která on sám do paměti počítače „propašoval“ a které vykonají zlomyslný kód. Tomuto kódu se říká shellkód. Vůči přetečení paměti jsou zranitelné programy napsané v jazycích překládaných přímo do jazyka procesoru (C, C++). Nejsou zranitelné programy napsané v moderních programovacích jazycích které běží na mezikódu (Java, C#).

7.1.3 Klíčové pojmy:

Přetečení paměti – zapsání více bytů do proměnné než má proměnná přiděleno
Zásobník – oblast v paměti počítače sloužící k ukládání dočasných (lokálních) proměnných, argumentů funkcí a návratové adresy
Návratová adresa – adresa instrukce, která následuje hned za instrukcí která zavolala právě prováděnou funkci. Až skončí právě prováděná funkce, procesor při vykonávání instrukcí skočí právě na tuto adresus
V jazyce C tato situace nastane například takto:

```
#include <stdio.h>
#include <string.h>

void main()
{
      char var[10];
      strcpy(var,"AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHH\n");
      printf(var);
}
```

Kód demonstrující jednoduché přetečení paměti

Do pole ve kterém je místo pro 10 znaků jsme jich umístili 32, tím došlo k přetečení paměti a přepsání místa které je rezervováno pro jinou proměnnou. Pokud toto přetečení nastane v hlavní funkci programu, dojde „pouze“ k narušení okolních dat. Pokud k němu ale dojde po zavolání funkce, můžeme získat kontrolu nad během programu. Je tomu tak z důvodu že pokud jsme v těle zavolané funkce, jsou na zásobníku uloženy speciální data, jejichž přepsáním lze dosáhnout změny běhu programu.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void spatna_funkce()    //0x0040100F
{
      char pole[10];
      //strcpy(pole,"AAAABBBBCCCCDDDD\x05\x10\x40\x00");
      strcpy(pole,"AAAABBBBCCCCDDDD\x05\x10\x40\x00");
      printf("\nPole: %s",pole);
}

void hacked()                 //0x00401005
{
      printf("\nFunkce ktera nebyla volana");
      getchar();
      exit(1);
}

int main(int argc, char *argv[])    //0x0040100a
{
      spatna_funkce();

      printf("\nAdresa funkce main:\t0x%08x",main);
      printf("\nAdresa funkce hacked:\t0x%08x",hacked);
      printf("\nAdresa funkce spatna_f:\t0x%08x",spatna_funkce);

      getchar();
      return 0;
}
```

Kód demonstrující získání kontroly nad během programu díky přetečení paměti a přepsání návratové adresy

Cíl přetečení paměti

Obecným cílem přetečení paměti je změna běhu programu v náš prospěch. Podle konkrétních potřeb se nejčastěji jedná o tyto varianty:

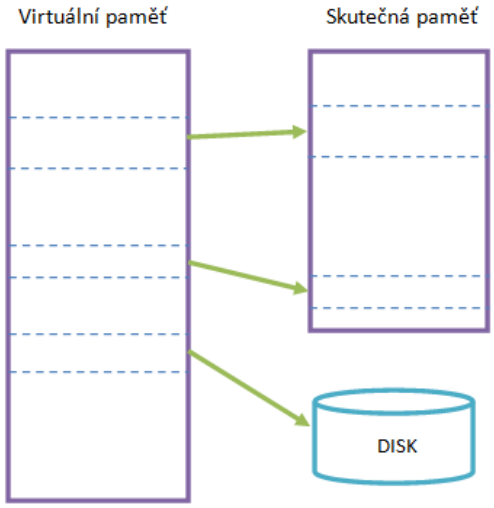
- odmítnutí služby: útočník chce vyřadit danou službu z provozu (konkurenční boj, podmínka pro provedení jiného typu počítačového útoku)
- nepovolený přístup k vzdálenému systému: útočník chce získat přístup a kontrolu nad vzdáleným systémem
- zvýšení privilegií: útočník má vzdálený nebo místní přístup na daný systém ale má příliš malé práva, chce je zvýšit nejlépe na úroveň root/administrator

Typy přetečení

- přetečení zásobníku: nejjednodušší typ přetečení, přepíšeme kontext funkce
- přetečení haldy: složitější typ, ne vždy jde zneužít
- útok na formátovací řetězce: objevený jako poslední, využívá chyby v architektuře funkcí používající formátovací řetězce

Paměť a adresování

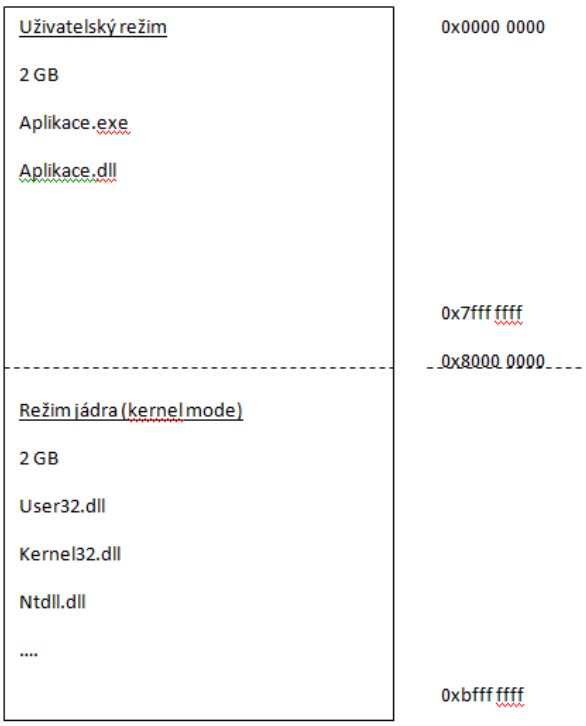
Když ve Windows spustíme aplikaci, nahrají se do paměti spustitelné soubory aplikace a její podpůrné knihovny. Každé aplikaci je přiřazeno 4 GB virtuální paměti, přestože v systému může být velmi málo fyzické paměti (např. 256 nebo 512 MB). 4 GB místa je založeno na 32 bitovém adresování (2^32 bitů odpovídá 4294967296 bytům). Když každá aplikace spustí manažer paměti, automaticky přiřadí virtuální adresy fyzickým adresám, kde se data skutečně nachází. Manažer paměti je záležitostí operačního systému a ne vysokoúrovňové softwarové aplikace – programátor se o ni tedy starat nemusí.



Obrázek 61: Virtuální paměť a stránkování

Paměť je rozdělena mezi uživatelský mód a mód kernelu. Paměť uživatelského módu je paměť kam se nahrává a spouští aplikace, zatímco paměť módu kernelu je paměť kam se nahrávají komponenty kernelu (tzn. jádra operačního systému) a kde se také spouští. Podle tohoto modelu by aplikace neměla být schopna přímého přístupu do paměti módu kernelu. Každý pokus o přístup aplikace do paměti módu kernelu má za následek hlášení

nepovoleného přístupu (access violation). Avšak v případě že aplikace potřebuje řádný přístup ke kernelu lze ji pomocí operačního systému přepnout.
Pro uživatelský mód a tedy pro každý program na Win32 platformě je k dispozici 2 GB virtuální paměti. Rozsah adres 0x00000000 až 0x7fffffff je pro uživatelský mód a rozsah 0x80000000 – 0xBfffffff pro mód kernelu.
Spustitelné soubory aplikace sdílejí adresní prostor nejen s DLL soubory aplikace (tedy se svými vlastními) ale také s DLL soubory operačního systému. Místo v paměti, kam se nahraje každý spustitelný nebo DLL soubor je stejné na všech počítačích se stejnou verzí operačního systému a stejnou verzí aplikace. To je velmi důležité při psaní exploitů – potřebujeme totiž znát přesné umístění DLL souboru v paměti a příslušných funkcí v paměti.



Obrázek 62: Struktura paměti procesu ve Windows

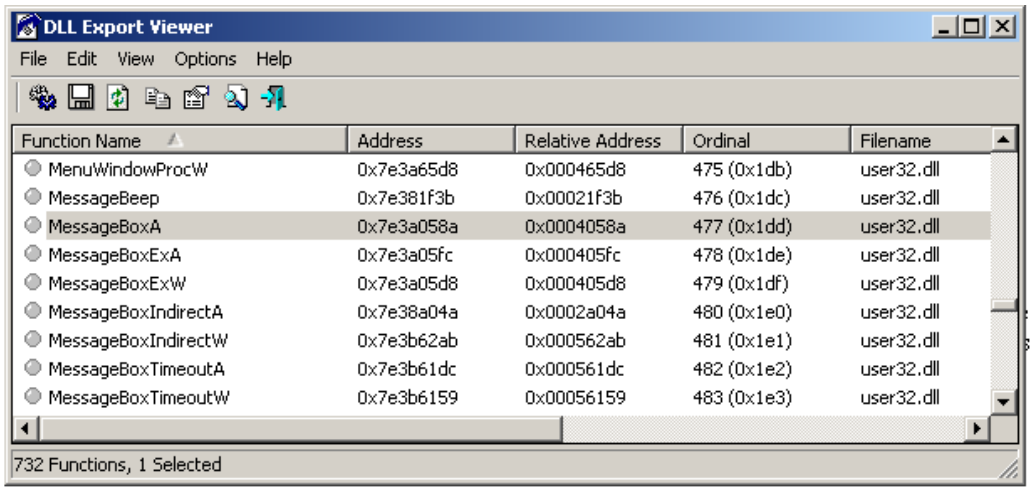
Nástrojů ke zjištění adresy kam se nahraje spustitelný soubor je dostupných mnoho, Microsoft poskytuje nástroj dumpbin.exe jako standardní součást programovacího prostředí Vistual Studio. Následuje výstup souboru kernel32.dll na systému Microsoft Windows XP Professional SP2 CZ.

c:\Program Files\Microsoft Visual Studio\VC98\Bin\DUMPBIN.EXE /headers kernel32.dll

```
Microsoft (R) COFF Binary File Dumper Version 6.00.8168
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

Dump of file kernel32.dll
PE signature found
File Type: DLL

FILE HEADER VALUES
             14C machine (i386)
               4 number of sections
        46239C40 time date stamp Mon Apr 16 17:54:40 2007
               0 file pointer to symbol table
               0 number of symbols
              E0 size of optional header
            210E characteristics
                   Executable
                   Line numbers stripped
```

```
                  Symbols stripped
                  32 bit word machine
                  DLL

OPTIONAL HEADER VALUES
             10B magic #
            7.10 linker version
           82200 size of code
           6DA00 size of initialized data
               0 size of uninitialized data
            B5AE RVA of entry point
            1000 base of code
           7F000 base of data
        7C800000 image base
            1000 section alignment
             200 file alignment
            5.01  operating  system  versionMicrosoft  (R)  COFF  Binary  File  Dumper  Version
6.00.8168
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.


Dump of file kernel32.dll
PE signature found
File Type: DLL

FILE HEADER VALUES
             14C machine (i386)
               4 number of sections
        46239C40 time date stamp Mon Apr 16 17:54:40 2007
               0 file pointer to symbol table
               0 number of symbols
              E0 size of optional header
             210E characteristics
                  Executable
                  Line numbers stripped
                  Symbols stripped
                  32 bit word machine
                  DLL

OPTIONAL HEADER VALUES
             10B magic #
            7.10 linker version
           82200 size of code
           6DA00 size of initialized data
               0 size of uninitialized data
            B5AE RVA of entry point
            1000 base of code
           7F000 base of data
        7C800000 image base
            1000 section alignment
             200 file alignment
            5.01 operating system version
(dále zkráceno)
```

Zvýrazněna je adresa, na kterou se nahraje do paměti kernel32.dll.
Můžeme také použít například volně dostupný program DLL Export viewer, který oproti dumpbin nabízí grafické rozhraní:



Obrázek 63: Nástroj DLL Export Viewer

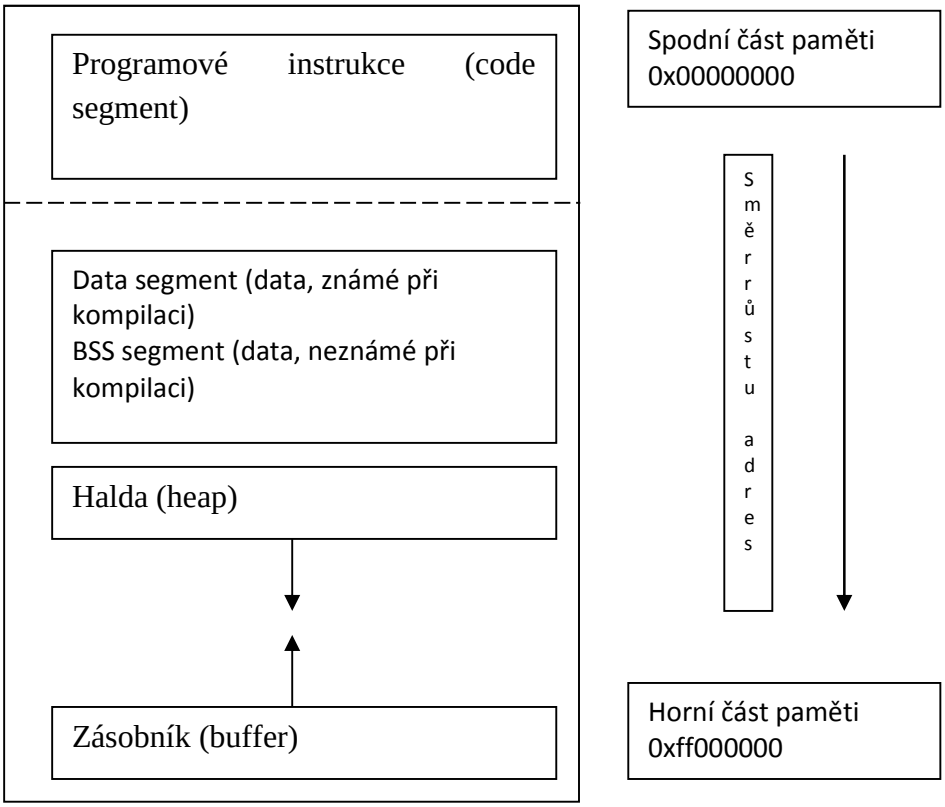
Struktura aplikace

Všechny aplikace mají svou virtuální paměť rozdělenu do těchto paměťových oblastí:

- segmentu kódu a textu (code, text)
- datový segment (heap, halda)
- BSS segment
- segment fronty (zásobníku, bufferu)

Segment fronty ukládá místní proměnné a volání procedur, datový segment uchovává statické a dynamické proměnné a textový segment uchovává instrukce programu.

Datový segment a segment fronty jsou soukromé pro každou aplikaci, to znamená, že žádná jiná aplikace nemá přístup do těchto oblastí. Textová část představuje naopak segment pouze pro čtení, ke které mohou přistupovat také jiné procesy. Avšak pokud dojde k pokusu o zápis do této oblasti, objeví se hlášení segment violation.



Obrázek 64: struktura paměti programu

Přidělování paměti – zásobník (buffer)

Zásobník roste směrem k nižším adresám. V zásobníku jsou uloženy:

- argumenty funkcí
- návratové adresy
- lokální proměnné funkcí

Přidělování paměti – halda (heap)

Halda roste směrem k vyšším adresám. V haldě jsou uloženy globální proměnné a dynamicky přidělovaná data (např. v jazyce C pomocí funkce malloc(), v C++ operátorem new).

```
//KOD01.CPP - ilustrace zakladni preteceni pameti. V tomto pripade jsou
prepsana pouze data
#include <stdio.h>
#include <string.h>

void main()
{
      char str1[]="AAAAAAA";
      char str2[]="BBBBBBB";
      char str3[]="CCCCCCC";

      printf("\n\nPrvni vypis. Vse je v poradku:");
      printf("\nstr1: %s",str1);
      printf("\nstr2: %s",str2);
      printf("\nstr3: %s",str3);

      //preteceni pameti: do promenne pro 7 bytu vlozime 15 bytu
      strcpy(str2,"XXXXXXX12345678");

      printf("\n\nDruhy vypis. Doslo k preteceni:");
      printf("\nstr1: %s",str1);
      printf("\nstr2: %s",str2);
      printf("\nstr3: %s",str3);

      printf("\n\nVypis  adres  promennych  v  hexadecimalnim  a  desitkovem
tvaru:");
      printf("\nstr1: %p (%i)",str1, str1);
      printf("\nstr2: %p (%i)",str2, str2);
      printf("\nstr3: %p (%i)",str3, str3);

      getchar();
}
```

Kód demonstrující přetečení paměti v haldě a v zásobníku

Procesor a vykonávání instrukcí

Programy napsané ve vyšších programovacích jazycích jsou kompilátorem přeloženy do jazyka konkrétního procesoru – operačního (nebo také binárního) kódu, který se skládá z jednotlivých instrukcí procesoru. Operační kód má podobu sledu čísel (nejčastěji v hexadecimálním nebo binárním vyjádření). Symbolicky lze binární kód vyjádřit pomocí tzv. assembleru – jazyka symbolických instrukcí který je přímou obdobou binárního kódu ovšem ve formě symbolického zápisu čitelného pro člověka (např. instrukce 0x90 procesoru x86 se v assembleru zapíše jako NOP, což je zkratka pro No Operation, prázdná instrukce). Při spuštění programu nahraje operační systém binární kód do paměti a poté jej procesor vykonává jednu instrukci po druhé tak jak jsou uloženy za sebou v paměti. Toto lineární vykonávání instrukcí je přerušeno pouze instrukcemi větvení (JNE, JE, JB,..) nepodmíněného skoku (JMP) nebo volání podprocedur (CALL). To která instrukce bude vykonávána jako

další v pořadí po tom co bude dokončena právě vykonávaná instrukce je pevně dáno adresou této instrukce která je uložena v registru procesoru EIP (Extended Instruction Pointer). Obsah tohoto procesoru nemůžeme přímo měnit.

Důležité instrukce jazyka assembler
PUSH – uloží obsah registru na zásobník
POP – vybere obsah zásobníku a uloží do registru
MOV – přesune obsah registru
JMP – nepodmíněný skok na adresu
CALL – volání procedury
NOP – žádná operace
CMP - porovnání
INC – zvýšení hodnoty
DEC – snížení hodnoty
ADD - sčítání
XOR – výlučný OR
AND – logické ano
OR – logické nebo
INT – volání přerušení
OUT , IN – vstupy a výstupy pro hardwarové porty

Důležité registry procesoru
EAX, EBX, ECX, EDX – obecné registry
EIP – instruction pointer
ESP – stack pointer
EBP – base pointer

Jak probíhá volání funkcí
Při volání funkcí jsou na zásobník uloženy nejprve parametry funkcí v obráceném pořadí v jakém jsou uvedeny ve volající funkci, poté je zde uložena návratová adresa (tzn. obsah registru EIP), po ní starý rámec zásobníku (tzn. obsah registru EBP) a poté je alokováno a použito místo pro lokální proměnné.

```
/*
STACK
     ----------------
     Local Variables
ESP->  i
     Buffer
     ----------------
EBP->  Old Value of EBP
     ----------------
     Return Address
     ----------------
*/
```

```
//KOD02 - (ve VC++ 6.0) stisknout F10 pro krokovani programu
//a ALT+8 pro zobrazení strojového kódu
//jednotlive radky krokovat bud pomoci F10 (preskakuje vnorene fce)
//nebo F11 (vstoupi do vnorenych fci)kod slouzi pro demonstraci jak
//pracuji registry EBP, ESP, EIP a jak se meni zasobnik pri volani funkce
```

```
#include <stdio.h>

int funkce(int a,int b,int c)
{
      char prom1[]="AAAAAAAA";
      char prom2[]="BBBBBBBB";
      char
prom3[]="DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDD";

      return a+b+c;
}

void main()
{
      int v;
      v = funkce(0x1111,0x2222,0x3333);
      printf("\n%i",v);
      getchar();
}
```

Čím přepsat návratovou adresu

Tím že dokážeme přepsat návratovou adresu uloženou v zásobníku a tím pádem ji vložit do EIP změníme běh program. Je jasné, že jej chceme přesměrovat na náš vlastní kód, problém ovšem je jak adresu našeho vloženého kódu zjistit. Musíme provést tzv. statickou analýzu – tj. zjistit konkrétní adresu zásobníku našeho programu. Tato adresa zůstává víceméně stejná pokud pustíme stejný program na stejné verzi stejného operačního systému. Pokud se změní (zranitelný) program, operační systém nebo jeho verze, bude adresa podstatně odlišná. Proto je pro počítačové útočníky tolik důležité zjistit přesnou verzi operačního systému a aplikace běžící na vzdáleném systému. I při dodržení těchto podmínek se ale může adresa zásobníku lehce lišit. Pro překonání krátkých odlišností existují dvě techniky: NOP sled a skok na registr.

NOP sled

NOP sled je nejstarší a nejrozšířenější technika pro úspěšné přesměrování přetečení zásobníku. Problém nalezení přesné návratové adresy řeší tím že zvyšuje oblast paměti do které se návratová adresa musí trefit tím že oblast před shellkódem zaplníme větším množství instrukcí NOP (0x90). Tato „vycpávka“ se nazývá NOP sled, protože pokud je návratová adresa přepsána jakoukoliv adresou uvnitř této NOP oblasti bude procesor vykonávat prázdné instrukce tak dlouho až nakonec dojde k samotnému shellkódu na jeho konci. Druhou možností je naplnit NOP instrukcemi oblast za shellkódem a na konec tohoto epilogu umístit instrukci relativního skoku která skočí samozřejmě na začátek shellkódu. Tato technika tedy útočníkovi ulehčí práci tím, že se stačí trefit do velké oblasti NOP sledu místo mnohem menší oblasti shellkódu.
Kvůli popularitě tohoto řešení ale mnoho IDS (intrusion detection system) vyhledává datové struktury s velkým množstvím NOP instrukcí. Nicméně je důležité si uvědomit že NOP-sled nemusí obsahovat pouze klasické NOP instrukce, může jít o jakékoliv instrukce, které nezmění stav počítače. Útočníci tedy na obranu proti IDS vyhledávajícím NOP vzorce začali NOP-sled vyplňovat náhodnými instrukcemi které neovlivní běh shellkódu.

Skok na registr
Při této technice využíváme jednak znalosti toho že někde v paměti se nachází instrukce skoku na konkrétní registr (např. JMP ESP) a jednak to že víme že v některém konkrétním registru je právě uložena adresa ukazující někam do zásobníku našeho programu. Pointou je přepsat návratovou adresu adresou takové instrukce čímž donutíme procesor skočit na adresu v daném registru – která ukazuje na námi kontrolovaný buffer a tudíž na shellkód. V praxi se využívá toho že instrukce které potřebujeme (např. JMP ESP, hexadecimálně FF E4) jsou součástí některé systémové knihovny která je nahrána při každém startu Windows a jejíž polohu tedy můžeme snadno určit. Opkód FF E4 se nachází například jeden byte od začátku instrukce call DbgPrint na adrese 0x7C941EED (na anglické verzi Windows XP SP2, jinde je adresa odlišná!). Takže pokud útočník přepíše návratovou adresu hodnotou 0x7C941EED (musí se ovšem nacházet na správné verzi Windows) donutí tím procesor vykonat instrukci zde uloženou – JMP ESP a tím spustit jeho kód. Tuto techniku velmi často využívají například internetoví červi.

Shellcode - shellkód
Shellkód je kód napsaný v assembleru přepsaný do podoby hexadecimálních čísel, který podstrčíme zranitelné aplikaci při přetečení paměti. Protože tento kód většinou musí projít funkcemi pro zpracování textového řetězce (např. strcpy()), nesmí obsahovat nulové byty (0x0), protože ty jsou těmito funkcemi považovány za ukončení textového řetězce.
Shellkód se liší podle toho pro jaký operační systém je určen. Obecně platí, že je těžší napsat shellkód pro Windows protože musíme znát absolutní adresy funkcí, které chceme v shellkód použít. V Unixových systémech můžeme použít systémová volání (např. int 0x80 v Linuxu) takže nemusíme znát přesné umístění funkce v paměti.
Co přesně má shellkód vykonávat je omezeno pouze fantazií a schopnostmi jeho programátora, může to být téměř cokoliv co lze v rozumě krátkém kódu v assembleru naprogramovat. Mezi nejčastěji prováděné akce patří nicméně tyto:

- získání lokálního shellu neboli příkazové řádky (odtud také název shellkód)
- otevření naslouchajícího socketu a jeho spojení se shellem
- otevření shellu a jeho spojení na vzdálený naslouchající socket (reverzní shell)

Obrana proti chybám přetečení paměti:

- volba bezpečného programovacího jazyka
- použití bezpečných knihoven (u nebezpečných jazyků)
- softwarová ochrana proti přetečení zásobníku
- ochrana paměťového prostoru s právy spouštění
- randomizace umístění systémových knihoven v adresovém prostoru

7.1.4 Internetové zdroje exploitů a databáze zveřejněných chyb

ICAT - http://icat.nist.gov
CERT.org - http://www.cert.org
Milw0rm.com - http://www.milw0rm.com
National Vuln Database - http://nvd.nist.gov
Open Source Vuln Database --http://www.osvdb.org
Secunia - http://www.secunia.com
SecuriTeam.com - http://www.securiteam.com
SecurityFocus Vulnerabilities - http://www.securityfocus.com
PacketStorm Security Exploits - http://packetstormsecurity.org
Securittem - http://www.securiteam.com/exploits/archive.html

Toto jsou veřejně dostupné zdroje, ve kterých jsou publikovány známé chyby a ne vždy ty nejaktuálnější exploity. Pro získání nejnovějších exploitů, případně 0-day chyb je nutné se ponořit do počítačového undergroundu. Žádný obecný návod neexistuje, v podstatě je nutné stát se členem některé hack komunity. Dobrý začátek může být účast na diskuzích různých IRC serverů a postupné propracovávání se hierarchií v dané skupině.

7.1.5 Případová studie – příklad napsání vlastního jenoduchého shellkódu ve windows

Jako základ vezmeme kód v jazce C – např. jednoduché zobrazení messageboxu (kod01.cpp). Kód přeložíme ve Visual C++ 6 (přirozeně lze i v jiných překladačích). Využijeme možnosti vývojového prostředí kód jednak krokovat (klávesa F10) a jednak při krokování zobrazit výsledný strojový kód (ALT+8). Tím můžeme získat buď assembler nebo přímo byte-kód (kod02.cpp). Kód můžeme otestovat například zpětným vložením do programu v jazyce C, jeho přetypováním na funkci a voláním (kod03.cpp).

kod01.cpp

```
//kod v jazyce C jako vzor pro shellkod
#include <windows.h>

void main()
{
      MessageBoxA(0,"HACK","HACK",0);
      ExitProcess(1);
}
```

kod02.cpp

```
//volani windows funkci z inline assembleru - MessageBoxA a ExitProcess
void main()
{
      __asm
      {
            xor         eax,eax     //null-byte pro ukonceni retezce
            push        eax
            push        0x4b434148  //KCAH - do zasobniku se ulozi opacne
            mov               esi,esp
            push        eax   //MB_OK
            push        esi   //adresa nadpisu
            push        esi   //adresa textu
            push        eax   //hwnd - 0=desktop
            mov               eax,0x7e3a058a
      //MessageBoxA(hwnd,text,nadpis,button) z user32.dll na winXPPro Sp2cz
            call        eax
            push        1
            mov               eax,0x7c81cdda
      //adresa funkce ExitProcess z kernel32.dll na win xp pro sp2 cz
            call        eax
      }
}
```

kod03.cpp

```
//Hotovy shellkod. MessageBoxA z user32.dll a ExitProcess z kernel32.dll na
win xp pro sp2 cz
void main()
{
      char
shellcode[]="\x33\xC0\x50\x68\x48\x41\x43\x4B\x8B\xF4\x50\x56\x56\x50\xB8\x
8A\x05\x3A\x7E\xFF\xD0\x6A\x01\xB8\xDA\xCD\x81\x7C\xFF\xD0";
      ((void (*)())&shellcode)();
}
```

Úkoly pro cvičení:

Úkol č. 1

Vytvořte program s následující strukturou

```
#include <studio.h>
void spatna_funkce()
{
      char pole[??]; //tyto dva radky upravte tak aby
      strcpy(????);  //doslo k přeteceni zasobniku
}

void fce1() { printf(„Funkce 1\n“);}
void fce2() { printf(„Funkce 2\n“);}
void fce3() { printf(„Funkce 3\n“);}

void main()
{
      spatna_funkce();
}
```

Cílem je upravit funkci „spatna_funkce“ tak aby se spustila některá z funkcí fce1 až fce4 bez toho aniž by jsme ji v programu explicitně volali (tzn. pomocí přetečení zásobníku).

Postup:

1.) zjistit adresy funkcí fce1() .. fce3()
2.) vytvořit si proměnnou pole o nějaké konkrétní velikosti
3.) zkopírovat do proměnné pole více bytů než pro kolik je alokováno místo
4.) zjistit které byty v kopírovaném řetězci způsobí přepsání návratové adresy (např. pomocí strcpy(pole,“AAAABBBBCCCCEEEEFFFFGGGG…“) – A má hexadecimální kód 0x41, B 0x42,..).
5.) nahradit tyto byty adresou některé z funkcí fce1()..fce3() (nezapomenou na to že jsou v paměti v převráceném pořadí „little endian", tzn. adresa 0x11223344 je v textovém řetězci psaná jako 0x44332211

Úkol č. 2

Pomocí krokování v některém z překladačů jazyka C/C++ (např. Microsoft Visual C++ 6.0) napište vlastní shellkód. Jako základ použijte jednoduché volání Win32 API funkcí (např. MessageBoxA nebo nebo WinExec – spuštění libovolného programu, třeba calc.exe).

Úkol č. 3

Vytvořte program, který vykoná následující shellkód

```
\x8B\xEC\x33\xF6\x56\xC6\x45\xF8\x63\xC6\x45\xF9\x61\xC6\x45\xFA\x6C\xC6\x4
5\xFB\x63\x83\xEC\x04\x6A\x01\x8D\x45\xF8\x50\xB8\x4d\x11\x86\x7c\xFF\xD0\x
6A\x01\xB8\xa2\xca\x81\x7c\xFF\xD0
```

! pozor tento shellkód bude fungovat pouze v konkrétní verzi Windows pro kterou byl vytvořený – Win XP Pro SP2 CZ.

Program bude mít následující strukturu:

```
#include <studio.h>
void spatna_funkce()
{
      char pole[??];
      strcpy(????);     //zde dojde k použití shellkódu
}

void main()
```

```
{
      spatna_funkce();
}
```

Postup:

1.) v překladači Visual C++ 6.0 je NUTNÉ smazat v Project/Settings/C,C++/Project options direktivu **/GZ** (jinak kompilátor k hotovému programu přidá kód kontrolující porušení zásobníku. Přetečení je možné i s tak ale je to složitější).
2.) Vytvořit proměnnou pole s dostatečným počtem bytů pro shellkód (tzn. nejprve je nutné kolik bytů má shellkód – ten lze zapsat jako řetězec char shellcode[]="\x90\x90…\xD0"; pak třeba funkce strlen() nebo operátor sizeof())
3.) zkopírovat do proměné pole více bytů než pro kolik je alokováno místo
4.) zjistit které byty v kopírovaném řetězci způsobí přepsání návratové adresy (např. pomocí strcpy(pole,"AAAABBBBCCCCEEEEFFFFGGGG…") – A má hexadecimální kód 0x41, B 0x42,..).
5.) zjistit adresu proměnné pole
6.) přepsat pomocí funkce strcpy() proměnnou pole tak že nejprve do ní vložíme shellkód a za něj adresu proměnné pole tak aby tato adresa přepsala návratovou adresu (tzn. dojde ke spuštění shellkódu)

„Trefení" se adresou proměnné na návratovou adresu lze udělat dvěma způsoby – buď manipulací s velikostí proměnné pole (zmenšování, zvětšování) nebo manipulací s velikostí shellkódu – přidáváním/odebíráním řetězce \x90 (= instrukce NOP, no operation) na začátek shellkódu.

Úkoly pro seminární práci

1. Vypracujte krátkou seminární práci (cca 1 A4 textu) na téma ochrany proti přetečení zásobníku v operačních systémech Windows (tzn. hlavně o technologiích DEP – data execution protection a ASLR – address space randomization layout a /GS přepínači kompilátorů Microsoft, dále ochrana systémových souborů proti přepsání atd.).

8 Bezpečnost účtů Windows

8.1 Systém hesel v OS Windows

Chcete-li používat moderní verze operačního systému Windows, musíte v něm mít zřízený účet. Při instalaci je automaticky vytvořen účet Administrator, ten ale obvykle na Windows XP a Windows Vista nemá možnost lokálního přihlášení. Další standardní účet je účet hosta Guest, ten také bývá zakázán. Dále jsou zřízeny servisní účty pro různé aplikace. Obvykle je to HelpAssistant (účet pod kterým se připojuje jiný uživatel při používání Vzdálené pomoci), dále pak IUS_jmenopocitace pro nainstalovanou službu IIS, ASPNET, SUPPORT_xxxxxxx, SQL atd. Účty, které si založíme při instalaci nebo při používání (například přes příkaz „control userpasswords2“ z příkazové řádky – tu spustíme například stiskem kláves Windows+R a napsáním „cmd“+Enter) se nazývají uživatelské účty. Můžeme jim přiřadit buď uživatelské oprávnění (omezené možnosti, např. nejdou instalovat programy apod) nebo správcovské oprávnění (téměř neomezené možnosti). Uživatele v systému si vypíšeme příkazem „net user“ z příkazové řádky. Přes tento příkaz jdou uživatelé na příkazové řádce i přidávat a rušit.

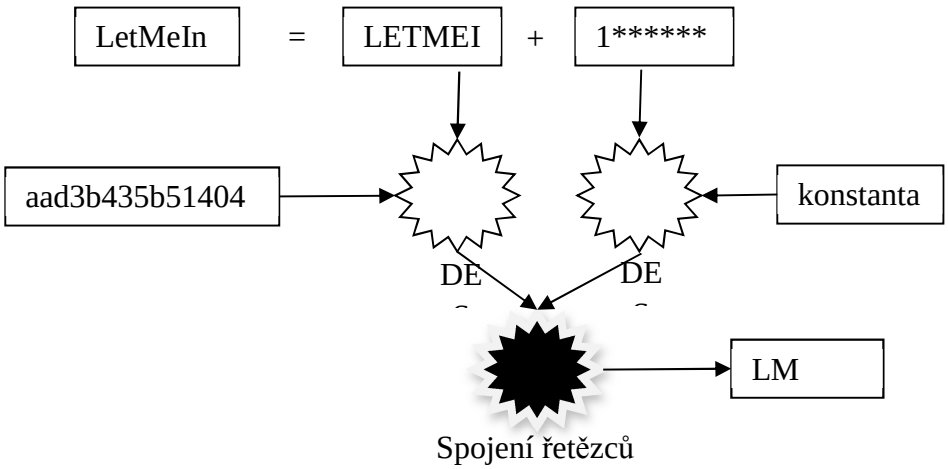
Každý účet ve Windows je spojen s heslem. Heslo není uloženo ve formě textu ale ve formě hashe, tedy jednosměrné kryptografické funkce. Konkrétně se toto úložiště nazývá Security Accounts Manager (SAM) a fyzicky jsou tyto data uloženy v registrech. Zkusit si vypsat tuto databázi si můžeme jednoduše z příkazové řádky příkazem „reg query HKLM\SAM“ případně pomocí nástroje regedit.exe – jak rychle zjistíme, nemáme zde přístup (chybová hláška „Error: Access is denied in the key HKEY_LOCAL_MACHINE\SAM\SAM“ nebo žádné zobrazené hodnoty) ani jako administrátor počítače. Jediný kdo zde má přístup je účet SYSTÉM, pod kterým se ovšem nemůžeme jednoduše přihlásit (Toto omezení lze obejít například tím, že na příkazové řádce zadáme příkaz „at 16:00 /interactive cmd.exe“ čímž vytvoříme naplánovanou úlohu která se spustí v námi zadaný čas a která vykoná spuštění příkazového řádku neboli cmd.exe. Vypsat naplánované úlohy můžeme přes příkaz „schtaskts“. Automaticky spuštěný příkazový řádek bude běžet právě pod identitou uživatele nt authority\SYSTEM, což si můžeme ověřit například právě přes příkaz „reg query HKLM\Security\SAM“ který by jinak vrátil chybu „Přístup odepřen“).

V registrech jsou tedy hash hodnoty našich hesel uloženy jako hodnoty klíčů. Jsou uloženy ve dvou různých formátech LM hash a NTLM hash. LM hash neboli také LAN Manager hash je verze jednocestné funkce která se používala ve verzích Windows až do verze Windows Vista (kde ji lze také použít, ale je nutné tuto volbu v systému ručně nastavit úpravou registrů) pro ukládání hesel které měly méně než 15 znaků. Až do verze Windows Me to byl jediný typ hashe který pro uložení hesel šlo ve Windows použít. V pozdějších verzích Windows byl tento typ zachován kvůli zpětné kompatibilitě. Bohužel až do operačního systému Windows XP SP3 včetně je používání LM hash defaultně povoleno což, jak si vysvětlíme dále, je značné bezpečnostní riziko.

SAM pracuje tak že pokaždé když vytvoříme heslo ve Windows32 systém jej prožene danou jednocestnou funkcí a výsledek uloží do SAM souboru: password -> 146b2ea8e7521b80ca48e185ddf92fe8

Přesný postup pro výpočet LM hash je následující:

1. heslo je zarovnáno NULL byty na přesně 14 znaků. Pokud je heslo delší než 14 znaků je zkráceno na 14 znaků
2. heslo je konvertováno do upper-case
3. heslo je rozděleno do dvou 7-bytových (56 bitů) částí převedených do bitových řetězců. Za každý 7. bit řetězce je vsunuta nula čímž je dosaženo délky 64 bitů což je délka potřebná pro DES klíče. Obě půlky jsou využity jako klíče k DES šifře.
4. každá část je použita pro zakódování pevného ASCII řetězce “KGS!@#$%” šifrou DES, tím dostaneme 2 8-bytové zašifrované texty (celkem 128 bitů)
5. dva výsledky z kroku 4 jsou zřetězeny a výsledek se nazývá LM hash



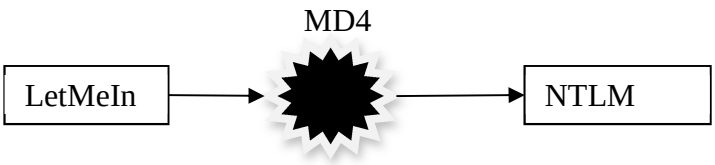
Obrázek 65: LM autentizace

Jak je na první pohled vidět, tento více jak 20 let starý algoritmus má hned několik kroků které snižují jeho kryptografickou účinnost, navíc samotná šifra DES je už považována za zastaralou. V OS Windows je pouze z důvodů kompatibility se softwarem a hardwarem který jej používá použít novější algoritmy. Algoritmus má hned dvě slabiny: hesla delší než 7 znaků jsou rozdělena na dvě části. Zdrojová abeceda pro zašifrování je navíc zkrácena tím že jsou znaky převedeny na upper case. Efektivně je tím šifra redukována z 95^14 možných hesell (pro heslo o maximální délce za využití všech znaků na US klávesnici) na 2^43 možných hesel. Moderní výkonné procesory dokážou za použití pouze alfanumerické sady znaků (jíž vyhovuje většina hesel) rozšifrovat LM hashe během několika hodin. Protože LM hash nevyužívá techniky solení (salt – viz. níže) lze použít time-memory tradeoff (zkrátit čas potřebný k prolomení šifry za cenu velkého využití před-počítaných hodnot), tzv. Rainbow Tables. Od roku 2003 je veřejně dostupný nástroj Ophcrack který díky před-počítaným Rainow tabulkám dokáže odhalit prakticky jakýkoliv LM hash založený na alfanumerické abecedě v řádu několika sekund. Další crackovací nástroje jako RainbowCrack, L0phtCrack a Cain nyní ve svých nových verzích nabízí podobnou funkcionalitu, čímž se prolomení LM hash stává triviální.

Microsoft v reakci na tyto bezpečnostní slabiny LM hash systému uvedl protokol NTLM. Windows až do verze Vista a Windows Server 2008 nicméně stále LM hashe využívaly. Windows Vista a Windows Server 2008 v sobě sice podporu LM hash stále obsahují, nicméně je po instalaci ve výchozí konfiguraci vypnuta a je nutné ji ručně zapnout přes Local Security Policy (Místní politika zabezpečení) v Administrative tools (nástroje pro správu). Nicméně tento nástroj není dostupný v edicích Home. NT hash se tedy na Windows Vista a Windows Server 2008 používá defaultně. Na Windows 2000, 2003 a XP se jako jediný použije defaultně tehdy, pokud zadáme heslo delší než 14 znaků.

NT hash se počítá následovně:

1. heslo je prohnáno funkcí MD4 (Message Digest verze 4) tím je získána hash



Obrázek 66: NTLM autentizace

Ani LM ani NTLM algoritmy nepoužívají techniku solení (salt). Solení (salting) je proces který byl poprvé použit v UNIXových operačních systémech před více jak dvaceti lety. Hashe hesel uživatelů v nich byly uloženy v textovém souboru který byl čitelný pro všechny. Pokud uživatel uviděl že jeho hash je stejn jako hash jiného uživatele zjistil tím, že mají stejné heslo. Proto tvůrci systému k heslu před jeho zasláním do hashovaní funkce přidali tzv. sůl (salt) která je pro každého uživatele

jiná a tím zaručili, že uživatelé se stejným heslem nebudou mít stejný hash. Zároveň se tím zvýšila kryptografická složitost hesla pro případný útok hrubou silou. Sůl z principu nebývá utajena, je proto spolu s hash uložena v textovém souboru v čitelné nezašifrované podobě.
Ve chvíli, kdy se uživatel přihlásí do Windows (jak již bylo řečeno, platí pro verze starší než Vista), jsou na jeho heslo použity **oba algoritmy** a **oba výsledky** jsou uchovány v paměti procesem LSASS (Local Security Authority Subsystem Service) a poté v registrech. Vnitřně je s heslem pracováno jako s 256-bytovým UNICODE řetězcem. Přihlašovací dialog je nicméně omezen na 127 znaků takže efektivně mohou uživatelé využít maximálně 127 znakové hesla. Účty programů, které se nepotřebují (a většinou ani nesmí) přihlašovat místně ale mohou teoreticky mít až 256 bytové hesla.

8.2 Napadení hesel

8.2.1 Sociální útok

Nejjednodušší způsob je kupodivu nejúčinnější – prostě se zeptat. Této technice se většinou říká sociální inženýrství. Podle jedné zahraniční studie z roku 2004 bylo 70% dotázaných osob ochotno sdělit své heslo do počítače neznámému člověku za tabulku čokolády. Taková situace je jistě do značné míry extrémní a lze se ptát nakolik to respondenti brali jako recesi, nicméně i tak platí, že člověk bude vždy nejslabším článkem počítačové bezpečnosti. Pokud se octneme v prostředí, které neznáme a jsme postaveni před problém (otázku, prosbu o pomoc, hledání řešení,,..) o němž nemáme dostatek znalostí, je téměř každý z nás náchylný k tomu nechat se zmást a zvláště pokud druhá osoba vzbuzuje sympatie, respekt nebo zdání oprávněné autority máme tendenci se nechal ovlivnit.
My se zde zaměříme ale spíše na čistě technické metody. Možností je několik:

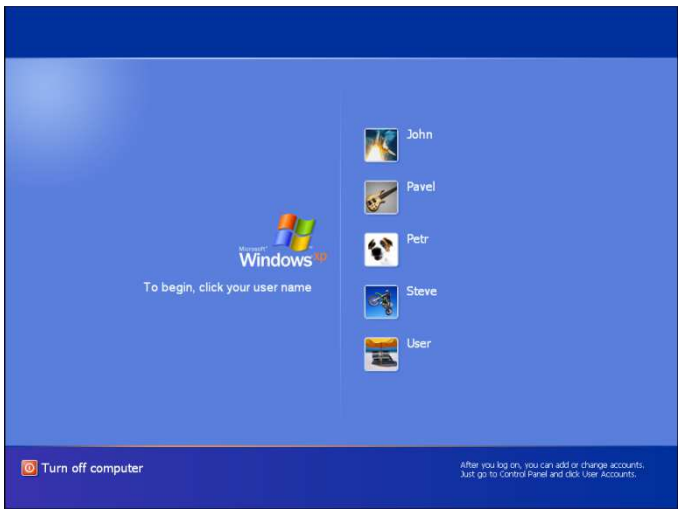
- zneužití zabudovaného účtu Administrator
- offline změna cizího hesla
- online útok na hashe
- offline útok na hase

8.2.2 Zneužití zabudovaného účtu Administrator

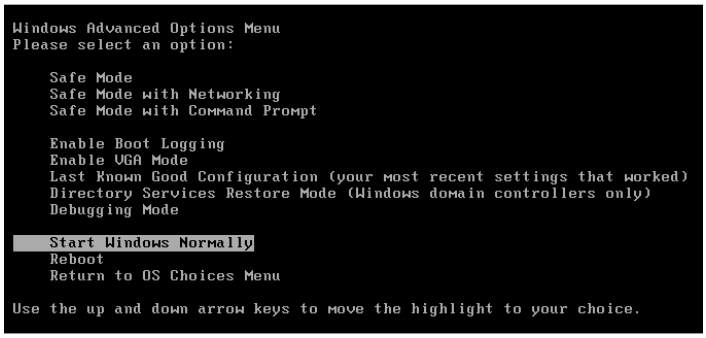
Operační systém Windows XP při instalaci vyžaduje zadání hesla pro účet Administrator (ve Windows Vista má účet Administrator defaultně zakázané místní přihlášení). Většina uživatelů toto heslo nezadává a zadají až heslo pro svůj účet pro jehož vytvoření je instalátor vybídne dále. Při normálním spuštění OS nám Windows účet Administrator nenabídnou (viz obrázek). Pokud přes kombinaci kláves CTRL+ALT+DEL vyvoláme přihlašovací okno do kterého se jméno účtu píše z klávesnice a zadáme Administrator, dostane se nám chybové hlášky, že se kvůli omezení účtu nelze přihlásit:

Obrázek 67: Zablokovaný účet administrátor

Pokud ale OS nabootujeme do bezpečného módu (safe mode) – při bootovací sekvenci podržíme klávesu F8 a v textovém menu zvolíme příslušnou položku, je nám umožněno se do systému přihlásit i pod tímto účtem. Počítače málo zkušených uživatelů jsou velmi často zranitelné touto chybou a jejich data jsou tedy přístupná i tehdy pokud mají zaheslovaný účet Administrator.



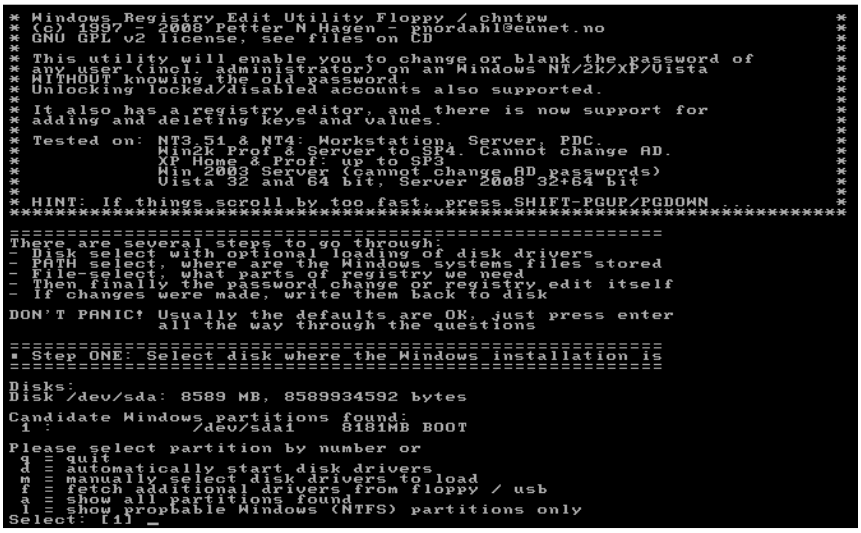
Obrázek 68: Přihlašovací obrazovka Window XP



Obrázek 69: Volba režimu bootování

8.2.3 Změna cizího hesla

Pokud má narušitel fyzický přístup k počítači, ale nemá čas nebo prostředky pro zjišťování hesel hrubou silou či rainbow tables má další možnost, i když poněkud brutálnější – změnit heslo offline. Opět stačí nabootovat (pokud je zaheslovaný BIOS je nutné fyzicky otevřít počítač a na 3 sekundy spojit jumper, který resetuje CMOS na defaultní hodnoty – má jej snad každý motherboard) z živé distribuce, tentokrát nazvané „Offline NT Password & Registry Editor“. Tento nástroj nám dovolí jednak měnit registry ale hlavně přímo hesla uživatelských účtů. Domovská stránka projektu je http://home.eunet.no/pnordahl/ntpasswd/. POZOR! Pokud některý z uživatelských účtů využívá souborový systém EFS (Encrypted File System) – souborový systém Microsoftu používaný pro realtimové šifrování souborů, dostupný od Windows Server 2003 nebude po změně hesla možné zašifrované soubory přečíst! Pokud si nejste jisti zda EFS používáte tak je téměř jisté že nikoliv.



Obrázek 70: Offline NT Password & Registry editor

Program pak sám přes textové menu dále provede jednotlivými kroky:

- Výběr disku (je možné nahrát i další ovladač např. z USB)
- Výběr umístění souboru s registry na disku (obvykle defaultní hodnota c:\windows\system32\config\
- Výběr části registry se kterou chceme pracovat
- Libovolná editace registru nebo
- Vypnutí SYSKEY ochrany (nedoporučuje se!! zneplatilo by hesla všech účtů) nebo
- Ovládání účtů: reset hesla na nulové heslo, změna hesla na nové heslo, povýšení práv účtu, odemknutí zamčeného účtu

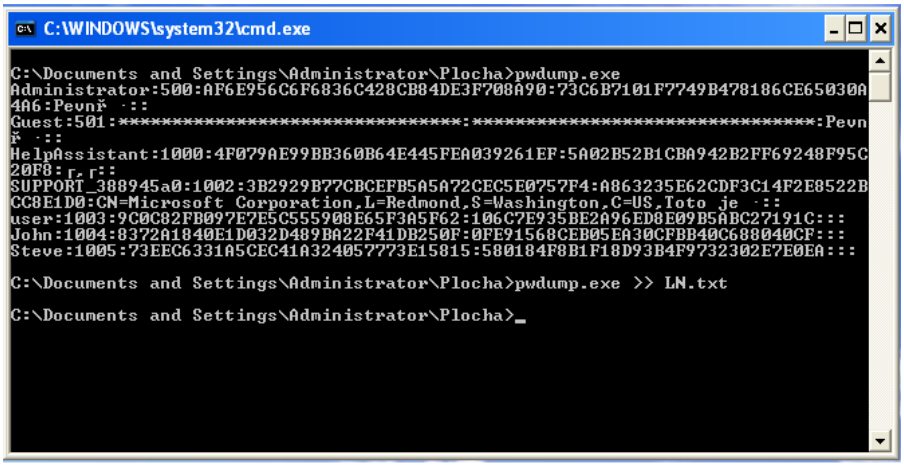
Obrázek 71: Finální obrazovka po změně hesla na „hacked“

Před ukončením se nás ještě program zeptá, zda chceme uložit změny. Poté následuje restart a máme volný přístup do systému. **Program nicméně nemusí pracovat vždy na 100%, proto spíše než změna hesla se doporučuje jeho reset na prázdné heslo** (potom si můžeme libovolné nastavit přímo ve Windows). **Může také dojít k nechtěnému pozměnění hesel u ostatních účtů** – sám autor přiznává že nástroj není zcela perfektní. Na druhou stranu je při použití tohoto nástroje stejně nevyhnutelné že vzbudí pozornost. Pokud použijeme analogii s fyzickým vloupáním určitě si všimneme že nám někdo změnil zámky v celém domu. Pokud je počítač součástí Active Directory domény, nedojde ke změně hesla do domény. Pouze ke změně hesla lokálního účtu. Hesla do domény jsou uložena úplně jinde – na doménovém řadiči v adresáři služby. Active Directory.

8.2.4 Online útoky

Pwdump.exe

Jinou možností je použití programu pwdump.exe – ten nám dokáže hashe vytáhnout i ze živého systému Windows prostým spuštěním:



Obrázek 72: Program pwdump

Na druhou stranu ovšem tento nástroj stejně musí být spuštěn pod účtem administrátora což na mnoha systémech ani není povoleno (což je správně). Spíše než k reálnému útoku tedy nástroj využijeme pro zkoumání vlastního počítače a testování systému.
Výstup z programu je formátu smbpasswd (používá Samba server na Unixech). Záznamy mají tuto strukturu:
<user>:<id>:<lanman pw>:<NT pw>:comment:homedir:
Kde <user> je uživatelské jméno na Windows NT, <id> je Windows RID (relative ID) = posledních 32 bitů z SID uživatele, <lanman pw> je LM hash hesla, <NT pw> je Windows NT (md4) hash. Pokud uživatel nemá nastavené heslo, je v záznamu 'NO PASSWORD*****'. Pokud je účet zablokovaný nebo neplatný je zde 32 znaků *.

8.2.5 Offline útoky

Jak již bylo řečeno, originální heslo není nikde uloženo. Pokaždé, když se pak do systému přihlásí uživatel je text hesla který zadá prohnán funkcí a výsledná hash porovnána s uloženou hash. Pokud chceme hesla uživatelů Windows „rozlousknout" musíme si tyto hash hodnoty z počítače nahrát a zkoušet pak slovníkovou nebo brute-force metodou prohánět hashovaní funkcí všechna hesla a porovnávat výsledky. To je velmi časově náročné a u delších hesel prakticky neproveditelné. Druhou možností, kterou máme je využití tzv. rainbow tables. Rainbow tables jsou speciální datová struktura která nám umožní „time-memory trade-off" nebo směňovat čas potřebný k vyzkoušení všech kombinací za místo v paměti. Raibow tables je potřeba vytvořit předem (na internetu jsou nejrůznější hotové rainbow tables k prodeji). Použitím RT se rychlost odhalování hesel zvýší v řádu stonásobků. Nejprve je ale nutné získat samotné hashe. Ve velmi starých verzích OS Windows bylo možné tyto hashe získat přímo, čímž byla práce hackerů usnadněna. V novějších SAM databáze chráněna jednak přístupovými právy a i po jejich obejití například nabootovaním do jiného operačního systému a připojením disku je nutné prolomit další vrstvu šifrování – tzv. SYSKEY. Konkrétní postup při použití oblibené hackerské distribuce linuxu Backtrack3 je následující:

1. nabootovat počítač z cd BackTrack3
2. Mount lokálního disku (obvykle je to hda1):
   mount /dev/hda1/
3. Extrakce SYSKEY potřebného pro dekódování SAM úložitě
   bkhive /mnt/hda1/WINDOWS/system32/config/SAM syskey
4. Nyní když máme SYSKEY můžeme extrahovat SAM soubor
   samdump2 /mnt/hda1/WINDOWS/system32/config/sam syskey>pwdhashe

Těmito kroky jsme úspěšně získali SAM soubor, nyní jej musíme cracknout. Soubor může vypadat nějak takto:

Administrator:500:NOPASSWORD**********:NOPASSWORD**********:::
Guest:501:NO PASSWORD**********:NO PASSWORD**********:::
HelpAssistant:1000:92641D9XXEF6C7781B2CD84185C101AB:8C21BE6FC1D84XXA353075731D911CF2:::

SUPPORT_388945a0:1002:NO PASSWORD*********************:9570ED1D667FD11AFE8BAD3FE7BB524F:::
user:1004:C6BEC284XX06B24CAAD3B435B51404EE:681C1C3190D8XX4C0FC0F7952D091DE2:::

První, čeho si všimneme, že je u účtu Administrator není nastaveno heslo (častá chyba, viz výše). Další záznamy jsou uživatelská jména (některá z nich jsou zabudované účty v systému, jiná vytvořená správcem pro uživatele) a hashe jejich jmen. Nyní je třeba použít některý z crackovacích nástrojů nebo rainbow tables. Populární crackovací nástroje pro Windows jsou například John the Ripper (http://www.openwall.com/john/) a Cain&Abel (http://www.oxid.it). My si ukážeme Rainbow Crack a Ophcrack

Rainbowcrack

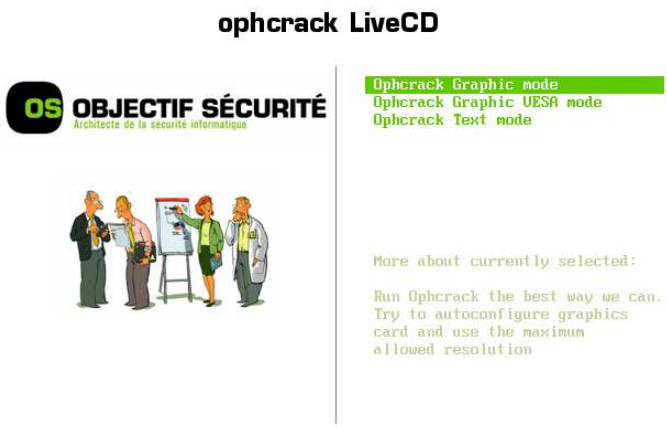
```
pwdump2 > dump.txt
rtcrack *.rt -p dump.txt
```

Prvním příkazem uložíme hashe do txt souboru (pro tuto akci musíme mít oprávnění administrátora). Druhým příkazem spustíme program který se pokusí hashe prolomit pomocí Rainbow tabulek (druhý parametr *.rt určuje že se použijí všechny soubory s příponou rt ve stejném adresáři jako je adresář aplikace rtcrack).

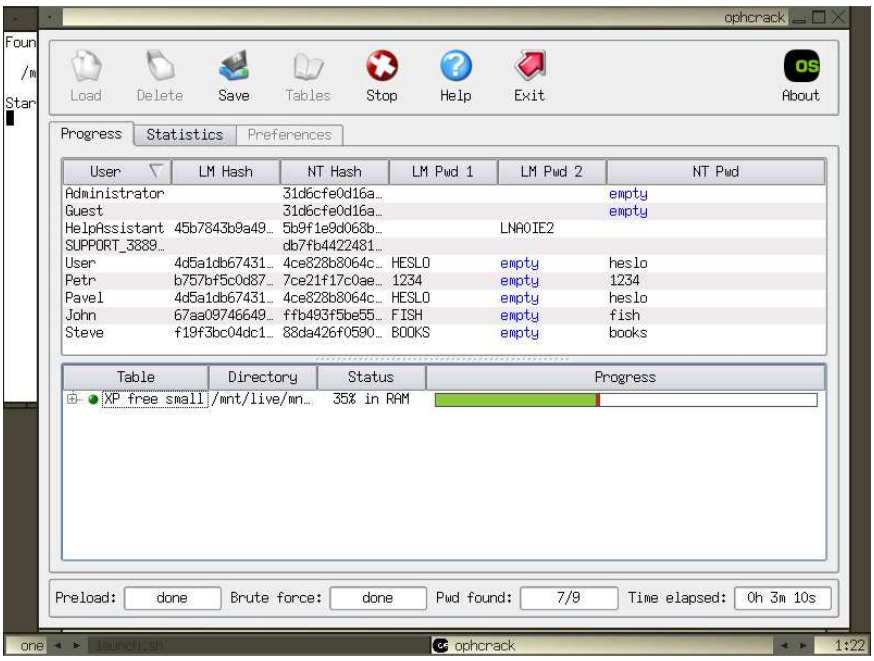
Nakonec jsem si nechal v současnosti zřejmě nejlepší nástroj pro zjištění hesel do Windows – Ophcrack.

8.2.6 Prolomení hesel do Windows účtů pomocí nástroje Ophcrack

Opthcrack je live-distribuce linuxu speciálně upravená pro jediný účet – prolamování hesel ze SAM databáze ve Windows. K jeho použití je nutný fyzický přístup k počítači, nabootování distribuce z cd. Nástroj pak automaticky mountuje disky, prohledává SAM úložiště a pokouší se odhalit hesla hrubou silou a pomocí rainbow tables. V závislosti na použité verzi (zdarma jsou ke stažení pouze základní) obsahuje různě rozsáhlé rainbow tabulky. Domovská stránka projektu je http://ophcrack.sourceforge.net.



Obrázek 73: Úvodní obrazovka proramu Ophcrack



Obrázek 74: Ophcrack v akci

Jak je vidět, nástroj dokázal během tří minut odhalit většinu hesel v počítači bez jakéhokoliv zásahu uživatele. V tomto případě nicméně byly použity jednoduché (i když nikoliv nereálné) hesla. Složitější hesla by nebyla nalezena kvůli nekompletní verzi Rainbow Tables. Kompletní Rainbow Tables lze na Internetu zakoupit. Skupina The Shmoo Group (http://rainbowtables.shmoo.com/) nabízí Rainbow Tables zdarma ke stažení přes BitTorrent. Kromě toho existuje několik webových stránek, které sice neposkytnou přímo Rainbow tables ale craknou zaslané hashe za použití jejich interních Rainbow tables.

Pokud LM hashe získáme jiným způsobem než fyzickým nabootováním a máme je uložené v souboru, můžeme Ophcrak použít k prolamování tohoto souboru. K tomuto účelu je také možné Ophcrak nainstalovat jako běžnou aplikaci do Windows.

8.2.7 Jak fungují Rainbow Tables

Výše uvedené programy fungují díky tzv. Rainbow tables. Následující text nám je představí trochu blíže. Předpokládejme, že máme k dispozici hash hesla a potřebujeme získat původní heslo. Smysl hashovacích funkcí je v tom že jsou jednocestné, tudíž není možné získat původní hodnotu analytickým způsobem. Můžeme pouze využít hrubou sílu. Máme několik různě chytrých možností jak konkrétně to provést:

- generujeme z dané abecedy, ze které je heslo vytvořeno (pokud nevíme přesně o jakou abecedu se jedná musíme zvolit dostatečně velkou množinu znaků aby bylo pravděpodobné že hledaná abeceda je podmnožinou naší abecedy) jednu kombinaci za druhou, každou kombinaci vložíme do hashovací funkce a výslednou hash porovnáme s naší hashí. Toto je nejhloupější způsob. I při nepříliš velké abecedě a délce hesla bude trvat extrémně dlouho. I když hash čirou náhodou najdeme, až budeme příště hledat jinou budeme muset celý proces zopakovat znovu.
- Opět z dané abecedy generujeme jedno heslo za druhým, každé heslo vložíme do hashovací funkce a výsledek i se zdrojovou kombinací uložíme do tabulky. Tabulku seřadíme podle hashí. Proces nalezení hashe se potom redukuje na nalezení položky v seřazené tabulce. Je to o něco chytřejší způsob, protože vyhledávání v seřazené tabulce je oproti neseřazené extrémně rychlé. Nicméně, musíme se smířit s tím, že prvotní generování tabulky bude opět trvat extrémně dlouho. Tomu se lze vyhnout použitím většího množství počítačů což v dnešní době není až tak velký problém. Hlavně ale uložení takovéto tabulky budeme už i malých hesel s malou abecedou potřebovat extrémní množství paměti. Při pevné délce hesla 8 znaků a zdrojové abecedě 26 malých písmen anglické abecedy přes 200 gigabytů,

přidáme-li i deset číslic, 2,8 terabytů. To je zhruba reálná hranice kdy si ještě tabulku můžeme uložit za použití běžných prostředků – a z dnešního hlediska je osmiznakové heslo z alfanumerických znaků malé anglické abecedy považováno za velmi slabé heslo! Zvětšíme-li heslo na deset znaků a přidáme i velké písmena abecedy dostaneme zhruba milion terabytů. Hesla dnes obvykle používají i speciální znaky a délka je u důležitých Windows systémů doporučena nad 14 znaků, tudíž je od jisté, z hlediska složitosti hesla poměrně nízké, složitosti nemožné se současnými technologiemi takovou tabulku uložit. Kdybychom tak měli nějaký způsob jak tuto obrovskou tabulku zmenšit..

- A právě k tomu slouží rainbow tables. Prvotnímu generování tabulky při které počítáme hash hodnoty ze všech hesel která je možné sestavit z dané abecedy o dané délce se bohužel stejně nevyhneme. Útěchou nám budiž to, že je to potřeba udělat pro každé heslo pouze jednou, navíc nejsme jediný kdo se o podobnou věc snaží takže není problém najít na Internetu komunitu sdílející vzájemně výpočetní výkon a jeho výsledky za tímto účelem. Tabulka samotná nám stále zabere hodně velké místo nikoliv již nereálně velké místo. Jak je to možné? Protože Rainbow tables (duhové tabulky, proč viz níže) nám dovolí směnit čas za místo. Dokonce si můžeme míru kompromisu mezi časem a velikostí dat sami zvolit! Nalezení 12-místného hesla o zmiňovaných parametrech je s rainbow tables v dnešní době otázka jednotek sekud, při velikosti tabulky v řádu desítek gigabytů. Čím menší máme tabulku, tím pomalejší bude vyhledávání (nicméně stále oproti dvěma výše popsaným způsobům extrémně rychlejší).

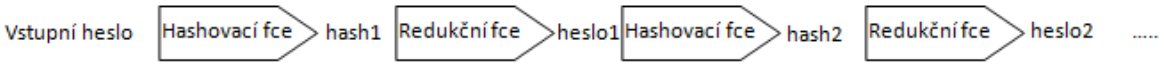
Pro přibližnou představu do jaké míry je reálné počítat hashe hrubou silou na současném domácím PC viz následující tabulka:
Tabulka: přibližné časy pro výpočet hash hodnot všech hesel z dané abecedy a délky hesla za předpokladu jeden milion vypočítaných hodnot za sekundu

Délka hesla	**6**	**8**	**10**	**12**	**14**
Délka abecedy					
10 (0-9)	**16 sekund**	**27 minut**	**44 hodin**	**0,5 let**	50 000 let
26 (a..z)	**1,4 hodiny**	**39 dnů**	71 let	48 000 let	3 miliardy let
36 (0-9,a-z)	**9,7 hodin**	**1,4 roku**	1800 let	2,4 milionů l.	3 biliony let
62 (0-9,a-z,AZ)	**10,5 dnů**	110 let	425 000 let	1,6 miliard l.	6,3x10^12 let
74 ↑ + další	**30 dnů**	456 let	2,5 milionu l.	1,3x10^10 let	7,5x10^13 let

Jak je z tabulky vidět, pokud má útočník k dispozici pouze běžný hardware a nikoliv například superpočítač, cluster pro distribuované výpočty nebo botnet nemá příliš šanci už ani u málo komplexních a krátkých hesel. Podobně nepříznivě by pro něj vyzněla tabulka zobrazující paměť potřebnou pro uložení takovýchto objemů dat. To se však při použití Rainbow Tables lehce obrátí.

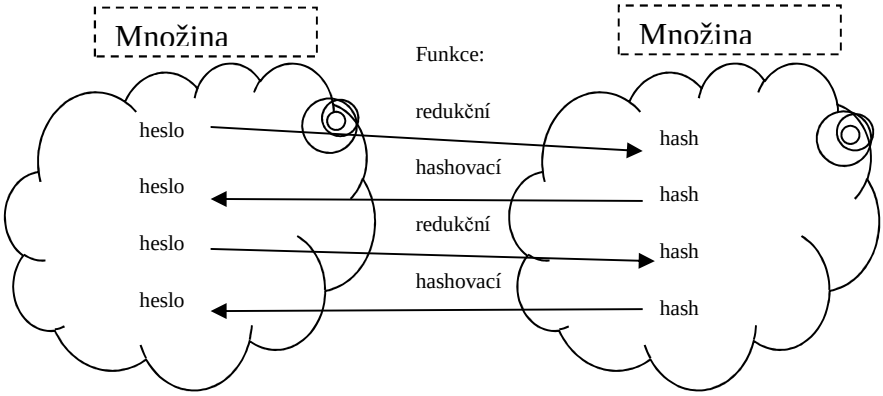
Jak tedy tato datová struktura, která umožnila v posledních třech letech obrovský pokrok v prolamování hesel uložených ve formě hashe, funguje? Nejprve se podívejme na to, jaká je vlastně její struktura a jak se vytváří:

1. Vygenerujeme slovník počátečních hesel nebo použijeme nějaký připravený. Pro každé heslo z tohoto slovníku použijeme následující postup:
2. Vstupní heslo vložíme do hashovací funkce. Výslednou hash vložíme do tzv. redukční funkce. Vstupem této funkce je hash, výstupem je text, jenž je vytvořený ze stejné abecedy a o stejné délce jako hledané heslo. Výstupem redukční funkce je tedy další heslo! To opět vložíme do hashovací funkce. Výslednou hash opět do redukční funkce, získáme další heslo a tak opět dokola, řádově až desetitisíc nebo i vícekrát.
3.



Obrázek 75: Rainbow řetězec

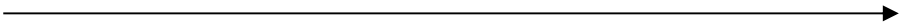
Po předem daném počtu kroků proces ukončíme a vezmeme pouze první (vstupní) hash a poslední hash (jiná varianta je uložit si poslední heslo) a uložíme je – tím získáme jeden řádek rainbow tabulky. Všechny hashe a hesla mezi nimi zahodíme. Tedy z řetězce „původní hash-heslo-hash-…-heslo-finální hash“ si uchováme pouze dvojici „původní hash-finální hash“. Je jasné že tím dosáhneme obrovské úspory místa. Celý řetězec lze také relativně rychle zrekonstruovat z původní hashe.



Obrázek 76: Hashovací a redukční funkce

Hashovací funkce mapuje množinu čistého textu na množinu hash hodnot, redukční funkce naopak. Tímto způsobem projdeme všechny hesla a tím získáme celou tabulku.

Vstupní heslo A	hash A1	heslo A1	hash A2	heslo A2	.....	hash AX	heslo AX	**Finální hash A**
Vstupní heslo B	hash B1	heslo A1	hash A2	heslo A2	.....	hash AX	heslo AX	**Finální hash B**
Vstupní heslo C	hash C1	heslo A1	hash A2	heslo A2	.....	hash AX	heslo AX	**Finální hash C**
Vstupní heslo D	hash D1	heslo A1	hash A2	heslo A2	.....	hash AX	heslo AX	**Finální hash D**
.....								
Vstupní heslo Y	hash Y1	heslo Y1	hash Y2	heslo Y2	.....	hash YX	heslo YX	**Finální hash Y**
Vstupní heslo Z	hash Z1	heslo Z1	hash Z2	heslo Z2	.....	hash ZX	heslo ZX	**Finální hash Z**



Postup výpočtu Rainbow Table

Rainbow table

Vstupní heslo A	Finální hash A
Vstupní heslo B	Finální hash B
Vstupní heslo C	Finální hash C
Vstupní heslo D	Finální hash D
.....	
Vstupní heslo Y	Finální hash Y
Vstupní heslo Z	Finální hash Z

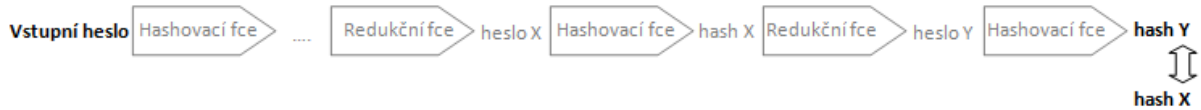
Uložený výsledek

Když máme Rainbow tabulku hotovou (některé jsou zadarmo ke stažení na Internetu, jiné se dají zakoupit) můžeme začít proces crackování:

- Vstupní hash ke které hledáme heslo porovnáme s každou poslední hashí uloženou v rainbow tabulce. Pokud ji v tabulce nenajdeme, vložíme hash do redukční funkce,čímž získáme nějaké textové heslo. Toto heslo vložíme do hashovací funkce a tuto hash opět porovnáme se všemi posledními hashi uloženými v tabulce. Tyto kroky opakujeme tak dlouho, dokud nenalezneme shodu mezi hashí kterou tímto procesem získáme a některou z finálních hashí uložených v tabulce.
- Ve chvíli, kdy najdeme shodu jsme nalezli řetězec „původní hash-heslo-hash-…-hash-heslo-finální hash“, ve kterém se nachází i heslo které hledáme! Protože však z celého řetězce máme uložený pouze začátek a konec, musíme jej ještě zrekonstruovat, což už je relativně nenáročná operace. V zrekonstruovaném řetězci si najdeme hash která se shoduje s naší původní hashí. **O jeden krok před ní je námi hledané heslo.** Celý proces si můžeme zobrazit na následujícím diagramu. Řekněme, že máme k dispozici **hash X**, která je součástí následujícího řetězce (řádku). Tím pádem tento řetězec obsahuje i hledané **heslo X**. Dopředu samozřejmě nevíme který je to řádek.

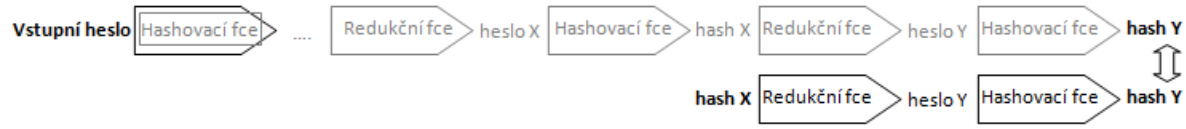
Řádek tabulky obsahující hash X

1. Krok



V prvním kroku porovnáme naši hash X s finální hashí (i v ostatních řádcích tabulky, to ale zobrazeno není). Shoda nenastala. Postupujeme do druhého kroku:

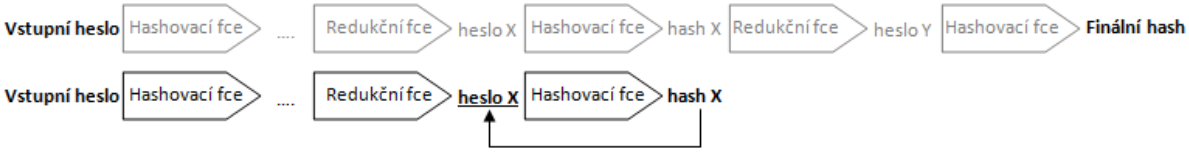
2. Krok



Hash X jsme vložili do redukční funkce, získali z ní heslo Y, to vložili do hashovací funkce a získali z něj hash Y. Shoda nastala (toto by byl samozřejmě téměř ideální případ kdy by hash X byla jen

kousek před koncem řetězce – čím blíže začátku hash X je, tím více kroků musíme udělat,než ji nalezneme)! Nyní víme, že naše heslo se nachází na tomto řádku. Musíme jej dopočítat - pozpátku to nejde, musíme postupovat od vstupního hesla, které máme uložené v prvním sloupci Rainbow tabulky. Počítáme tak dlouho, dokud nenarazíme na ***hash X****. Jeden krok před ní najdeme* ***heslo X****.*

3. Krok



V závěrečném kroku dopočítáme řetězec až k hash X a získáme tím heslo X.

Nyní si povězme něco více o redukční funkci. Redukční funkce tedy dělá do jisté míry opačnou věc než hashovaní funkce – z hash hodnoty udělá hesla. Ale pozor! V žádném případě tato funkce nedokáže hash obrátit a zjistit z jakého hesla byla vytvořena – to totiž není možné. Prostě jen na základě nějakého deterministického postupu mapuje množinu hashí na množinu čistého textu. Zajímavé na redukčních funkcích je to, že můžou být do jisté míry libovolné, nezáleží na tom, jakou redukční funkci použijeme, stačí že splňuje ten předpoklad že generuje hesla o dané délce a dané abecedě (tím pádem nemůžeme zaměňovat například redukční funkci pro hesla o délce 8 znaků složených jen z čísel za redukční funkci pro hesla o deseti alfanumerických znacích).

Hashovací funkcí (musí být stejná jaká byla použita k vytvoření námi zkoumané hashe) převedeme heslo (označované také jako plaintext, čistý text) na hash. Poté tzv. redukční funkcí z hashe vytvoříme opět plaintext o stejné délce. Jak již bylo řečeno, redukční funkce může být libovolná. Může to být například i něco tak jednoduchého jako je funkce, která prostě vezme prvních šest znaků z hashe (pokud by souhlasila abeceda hashe a našich hesel). Výsledný plaintext opět proženeme hashovací funkcí. Výsledek zredukujeme na nový plaintext. Opakujeme pro libovolný počet kroků – toto je parametr, který si sami nastavujeme a který má vliv na time-space trade-off čili směnu času za místo. Čím více kroků tím menší bude výsledná tabulka, ale déle trvá její procházení při crackování.

Např. vstupní heslo **123456** (vstupní abeceda alfanumerické znaky malé anglické abecedy)
MD5(**123456**) = e10adc3949ba59abbe56e057f20f883e
Red(e10adc3949ba59abbe56e057f20f883e) = e10adc
MD5(e10adc) = 96bf38d01b84aa16cf2bb9f55c61ac85
Red(96bf38d01b84aa16cf2bb9f55c61ac85) = 96bf38
….
MD5(068696)= **e3225ab333b37defddba2a8022c0c468** (finální hash)
Do rainbow tabulky si zapíšeme pouze počáteční plaintext a konečnou hash:
12345: e3225ab333b37defddba2a8022c0c468

Tímto způsobem dokážeme prohledat obrovský seznam hashí které vlastně nemáme nikde uloženy. Variantou tohoto algoritmu je že se neukládá finální hash ale finální plain-text.

Problémem u takto předpočítaných řetězců hashí jsou kolize. Obecně je u hashovacích algoritmů považováno za velmi nežádoucí vlastnost pokud produkují příliš mnoho kolizí. Ironicky je takový hashovací algoritmus bezpečnější proti útoku s rainbow tables. Kolize hashí totiž v rainbow tabulce způsobují jednak zacyklení (v případě kdy více hashí generuje stejný plaintext) a jednak konvergenci více plaintextů do jedné hashe (v případě kdy více plaintextů má stejnou hash). Tyto větve je nutné z tabulky vyndat, čímž ovšem přijdeme o určitý počet hashí a jejich plaintextů. Rainbow tables se kolizím a cyklům brání tak, že každý sloupec je kódovaný jiným redukčním algoritmem.

Rainbow Tables obvykle ukládají řádově 1 hash z tisíců až desetitisíců. Čas potřebný pro crackování se s RT snižuje se čtvercem paměti, kterou máme k dispozici. Tzn. dvojnásobně velká tabulka zvýší rychlost procházení čtyřikrát. Algoritmus pro redukční funkci může být libovolný, nicméně pro hesla o různé délce a různých abecedách se samozřejmě liší (šestimístné heslo složené z písmen bude generováno jinou funkcí než dvanáctimístné s alfanumerickými znaky).

Obrana proti Rainbow Tables spočívá v použití soli (salt). Microsoft Windows jsou jednou z jedním z mála operačních systémů které sůl nevyužívají. Navíc má jejich implementace LM chybu, díky které se heslo nepočítá doopravdy ze 14 znaků ale jako 2 sedmiznakové hesla, převedené navíc na velké písmena. Tím se snižuje náročnost z 2^84 na 2^36 kombinací. Zhruba do náročnosti 2^30 se dá s dnešní výpočetní technikou použít prostá hrubá síla, zhruba od 2^50 je již kombinací příliš mnoho i pro použití Rainbow Tables (jejich sestavení by trvalo příliš dlouho). Windows hesla uložená pomocí LM s náročností 2^36 tedy spadají právě do zranitelného okna. Používání LM hashí lze ve Windows XP a Windows 2003 vypnout. Ve Windows Vista jsou vypnuty defaultně.

8.2.8 Online Rainbow Tables

Pokud útočník nechce provádět crackování získaných hash sám pomocí programů popsaných výše (Ophcrack, Rainbow crack) může využít online služeb:
http://loginrecovery.com: Tato stránka má zřejmě nejkvalitnější (nejrozsáhlejší) tabulky. Je zde také ke stažení bootovací cd nebo disketa, s jejichž pomocí se hashe dají získat obdobně jako s jinými nástroji. Získané hodnoty se uploadují přímo na stránce. Do deseti minut jste informování jestli hash jde nebo nejde cracknout. Poté můžeme buď zaplatit (30 dolarů) pro okamžité ukázání hesla nebo počkat 72 hodin, po této době bude heslo ukázáno zadarmo (doručeno emailem).
Loginrecovery dokáže odhalit většinu hesel (rainbow tables nikdy nezaručují 100% úspěšnost) do délky 32 znaků skládající se z abecedy „ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !@#$%^&*()-_+=~`[]{}|\:;"'<>,.?/”. Server má k dispozici 80 GB tabulky které lze prohledat právě do deseti minut. Pokud nechceme ani platit ani čekat 48 hodin a hledané heslo není tak složité můžeme využít jiné stránky.

Obrázek 77: Služba LoginRecovery

Existují i další stránky poskytující tyto služby – dokonce zadarmo, na druhou stranu ale zřejmě nemají k dispozici tak kvalitní Rainbow tables. Jsou to například:
http://plain-text.info/
http://astalavista.com
http://LMcrack.com

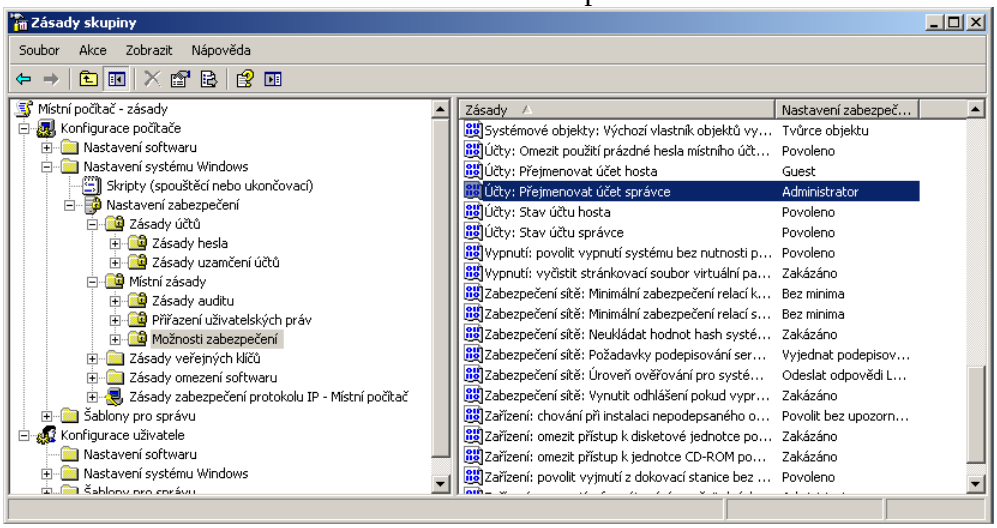
MAC OS X 10.3

MAC OS X 10.3 (Panther) k ukládání hesel také používá LM a NTLM hashe. Postup jejich cracknutí je úplně stejný jako v případě Windows hashí. Hashe jsou uloženy v souboru /var/db/shadow/hash/<generateduid>. Jsou 104 bytů dlouhé, 64 bytů mají zřetězené NTLM+LM hashe a 40 bytů SHA1 hash. Hodnotu <generateduid> pro daného uživatele zjistíme příkazem „niutil -readprop . /users/<username> generateduid“. Prvních 64 bytů z uloženého souboru získáme příkazem „sudo cut –c1-c64 /var/db/shadow/hash/<generateduid>“ (cut je program pro vyseknutí části řetězce, sudo kvůli právům). Hashe jsou uloženy v pořadí NTLM hash a hned za ní LM hash, tedy v opačném pořadí než v pwdump formátu. Proto je před předáním do nástrojů určených pro Windows nutné přehodit dvě 32-bytové půlky a vložit mezi ně dvojtečku. V MAC OS X 10.4 (Tiger) bylo ukládání hesel lépe zabezpečeno. Nicméně pokud má uživatel zapnuto sdílení souborů a složek Windows, jsou opět použity NTLM+LM hashe a útok lze provést.

Obrana

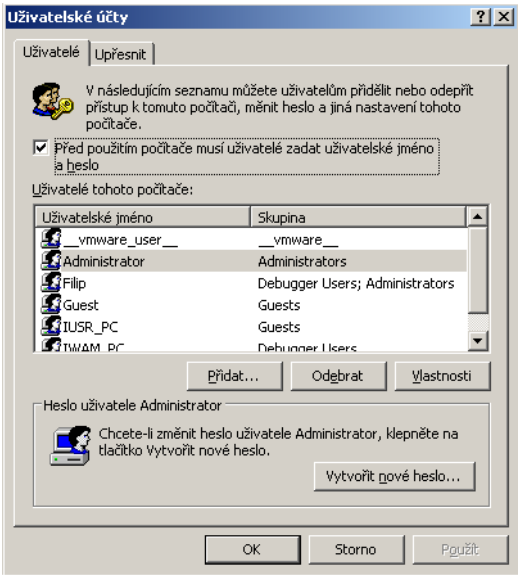
Možnosti obrany jsou následující:

- při instalaci Windows zadat heslo pro účet správce,
- zakázat v BIOSu bootování z CD-ROM a USB, zaheslovat BIOS
- využít možnosti přejmenovat účet správce: z příkazové řádky spustíme nástroj Zásady skupny (Group Policy Editor) příkazem gpedit.msc (pokud je náš počítač součástí domény je postup odlišný). Vybereme větve Konfigurace počítače/Nastavení systému Windows\Nastavení zabezpečení\Místní zásady\Možnosti zabezpečení a zde položku „Účty: Přejmenovat účet správce“. Zadáme libovolné jméno. Tímto nástrojem je možné definovat celou řadu dalších bezpečnostních nastavení počítače.



Obrázek 78: Zásady skupiny

- Pokud používáme Windows XP Home Edition, gpedit.msc není dostupný. V tom případě zadáme v příkazovém řádku „control userpasswords2“ a otevřeném dialogu nastavíme Vlastnosti účtu Administrator, konkrétně jeho jméno.



Obrázek 79: Nástroj uživatelské účty

8.2.9 Bezpečná hesla

Systém ukládání hesel ve Windows má svoje slabiny ale v zásadě bezpečný je. Musíme však zvážit následující zásady při tvorbě hesel:

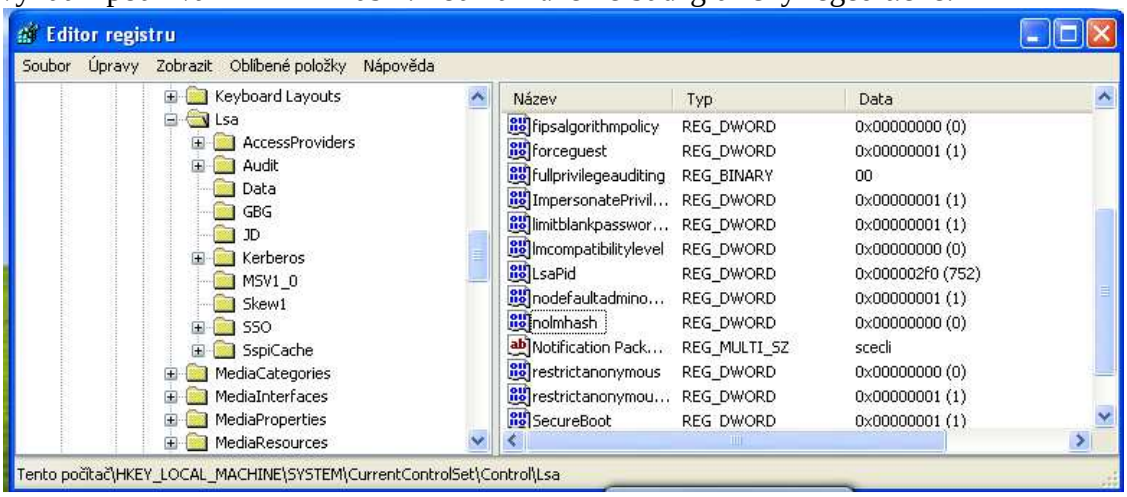
1. Dlouhé hesla: dobré heslo by mělo být nejméně 8 znaků dlouhé a čím delší tím lepší. Hesla která jsou kratší než 8 znaků jsou při dnešní výpočetní síle běžně dostupných počítačů málo odolná proti prolomení hrubou silou. Za předpokladu že útočník dokáže provést 3.000.000 pokusů nad zachyceným LM hashem za sekundu a heslo je zcela náhodně sestaveno ze všech 69 znaků (včetně mezery) US klávesnice a má délku 8 znaků, zabere vyzkoušení všech možností 3 roky (přirozeně útočník může mít štěstí a trefit se napoprvé. Jde o statistické údaje). Pokud je použita modernější NTLMv2 autentizace odhaduje se že je možné dosáhnout pouze 800.000 pokusů za sekundu čímž se doba pro vyzkoušení všech možností protáhne na deset let. Tyto kalkulace jsou nicméně prováděny na základě výpočetní síly současných počítačů. Pokud vezmeme v úvahu Moorův zákon, zabere nejpozději za pět let cracknutí LM verze hesla o osmi znacích pouze 98 dnů místo tří let. Za pět let tedy bude muset být použito heslo aspoň o devíti znacích k tomu aby heslo odolalo průměrně 180 dnů. Z hlediska kombinatoriky je mnohem důležitější délka hesla než délka zdrojové abecedy.
2. Komplexní hesla: dobré heslo se skládá ze všech čtyř dostupných typů znaků (nepočítáme-li znaky národních abeced): malé písmena, velké písmena, číslice a ne-alfanumerické symboly.
3. Heslo často měnit: na základě toho jak cenný systém heslo chrání by se mělo měnit 1x až 4x do roka. Tyto hodnoty vychází z propočtů současné výpočetní síly vs. výpočtů nutných pro prolomení současných algoritmů pro hesla o doporučené délce (8 znaků)
4. Použít heslo pouze na jediném místě: heslo je pouze tak bezpečné jak bezpečný je nejméně zabezpečený systém ve kterém ho použijeme
5. Používat pouze jedinou osobou: sdílení hesla s kýmkoliv jiným (v organizacích se často používají „firemní“ hesla) dramaticky snižuje jeho bezpečnost
6. Nepoužívat heslo na nedůvěryhodných počítačích: toto se týká zejména veřejných počítačů v knihovnách, internetových kavárnách, hotelech,.. Heslo lze na takových místech odposlechnout i bez toho že by útočník měl Administrátorský přístup do počítače. Hardwarové keyloggery jsou dnes již velmi levné a nenápadné (např. http://www.keyghost.com).

Proč se vůbec ještě LM hash používají, když jsou tak nebezpečné? Jednak jsou tak snadno napadnutelné až v poslední době a jednak z důvodů kompatibility se staršími zařízeními. Pokud se

útočníkovi povede získat LM hash nebo NT hash (ať už offline nebo z živých Windows) v podstatě se crackováním hesla nemusí ani zabývat. Tato hash samotná totiž naprosto stačí k autentifikaci uživatele např. v síťových službách. V tomto smyslu tedy není rozdíl mezi 1-znakovým heslem uloženým pomocí LM a heslem se 127 znaky s vysokou složitostí uloženým pomocí NTLM. Pokud technicky zdatný útočník získá kteroukoliv tuto hash má vyhráno. Tento typ útoku se nazývá „Pass The Hash".

8.2.10 Jak zakázat LM hashe ve Windows XP

LM hashe jsou v současné době jednou z největších zranitelností OS Windows (viz. výše). Na Windows Vista už jsou defaultně zakázány, na dřívějších verzích (2000, XP, 2003) bohužel ne. Použijí se k uložení hesla vždy, když je naše heslo kratší než 14 znaků (v opačném případě se použije bezpečnější NTLM hash). Používání LM hashí lze v těchto OS naštěstí zakázat ručně. Ve Windows XP je postup následující: v registru najdeme větev HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa a v ní klíč nolmhash. Pokud zde tento klíč není, přidáme jej (typ REG_DWORD). Pak nastavíme hodnotu na 1. Tímto jsme si vynutili používání NTLM hashí. Použít můžeme buď grafický regedit.exe:



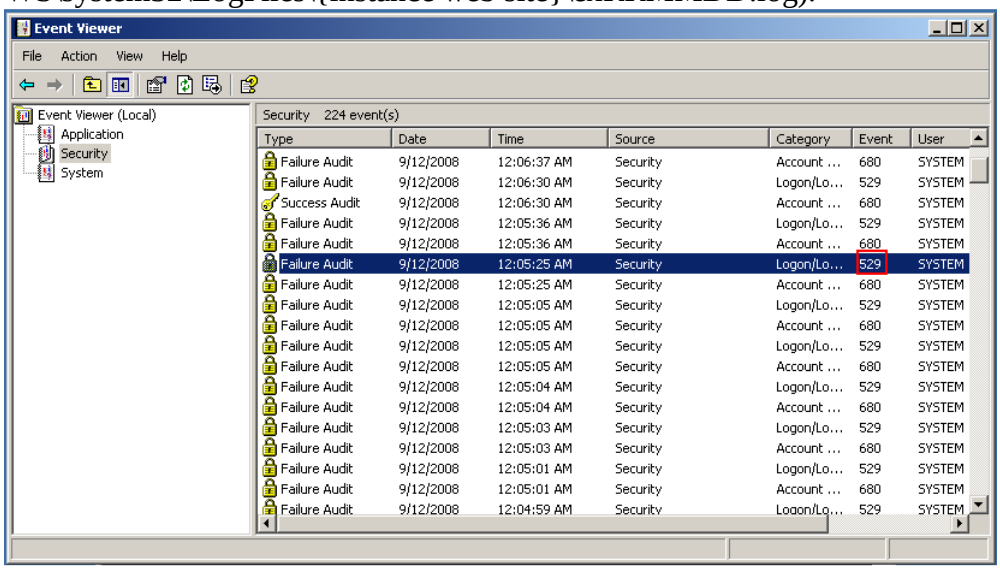
Obrázek 80: Nástroj editor registru

Nebo z příkazové řádky
„reg add HKLM\SYSTEM\CurrentControlSet\Control\Lsa\ /v nolmhash /t REG_BINARY /d 1 /f"
Pozor! Při příštím nabootování Windows se sice už nebudou defaultně využívat LM hashe, nicméně jejich stávající hodnoty zůstanou v registru až do doby než si uživatel heslo změní – pak už bude uložená pouze NTLM hash. To však nic nemění na faktu že pokud má uživatel slabé heslo, neuchrání ho ani NTLM hash před jeho odhalením.

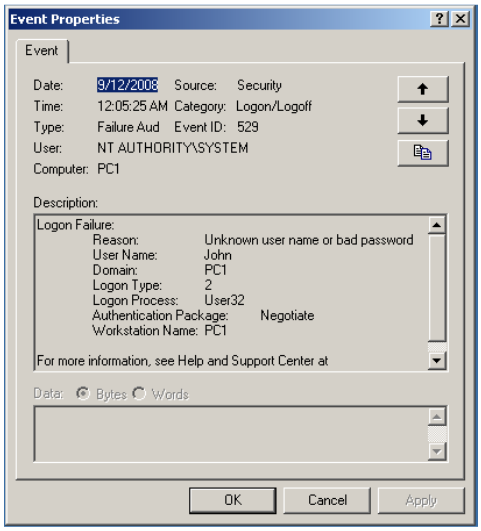
8.2.11 Kontrola logů

Pokud už prolomení hesel nedokážeme zabránit je alespoň dobré vědět o tom že k němu došlo. Operační systém Windows, stejně jako každý jiný slušný operační systém, vede záznamy o důležitých událostech v tzv. logu. To které události se mají sledovat se dá nastavit jednak na úrovni operačního systému a jednak na úrovni jednotlivých aplikací. Mezi sledované akce samozřejmě patří i přihlášení uživatelů – ať už úspěšné nebo neúspěšné – a jejich odhlášení.
Kontrol logů je činnost kterou by měl pravidelně provádět každý správce počítače nebo sítě. OS Windows obsahuje tři základní skupiny logů – Aplikace, Zabezpečení, Systém (v anglické verzi Application, Security, System). Mohou zde být i další logy, podle toho jaký software je nainstalován případně jakou roli plní server – běžné jsou např. Antivir, DNS, DHCP, Exchange. Do logů se zaznamenávají tzv. události. U každé události je zaznamenáno datum, čas, typ (informace, upozornění chyby, ne/úspěšné provedení operace), uživatel který událost vyvolat, identifikace zdroje události (počítač, IP adresa, proces), kategorie (liší se podle toho ve kterém jsme logu, v Zabezpečení například Přihlášení/odhlášení) a EventID. EventID je jakási číselná klasifikace

jednotlivých typů událostí, podle ní lze často najít na Internetu podrobnosti ke konkrétnímu hlášení. Dobrým zdrojem informací o událostech je server http://www.eventid.net/ . Z hlediska bezpečnosti nás zajímá především událost logu Zabezpečení s **EventID 529**. Tato událost je do logu zapsána vždy když dojde k neúspěšnému pokusu o přihlášení do počítače – ať už fyzicky na konzoli (někdo zkouší hádat hesla na klávesnici – v logu bude uveden název stanice) nebo vzdáleně například přes vzdálenou plochu (v logu bude uvedena IP adresa) nebo třeba v rámci nějakého typu přihlašovacího procesu na webovém serveru (v takovém případě nebude v popisu události v logu uvedena IP adresa nebo název počítače ale ID procesu. Z jaké IP adresy útok přišel je potom nutné vysledovat v logu webového serveru – v případě IIS se tento nachází v C:\WINDOWS\system32\LogFiles\{instance web site}\exRRMMDD.log).



Obrázek 81: Nástroj Event Viewer



Obrázek 82: Vlastnost události

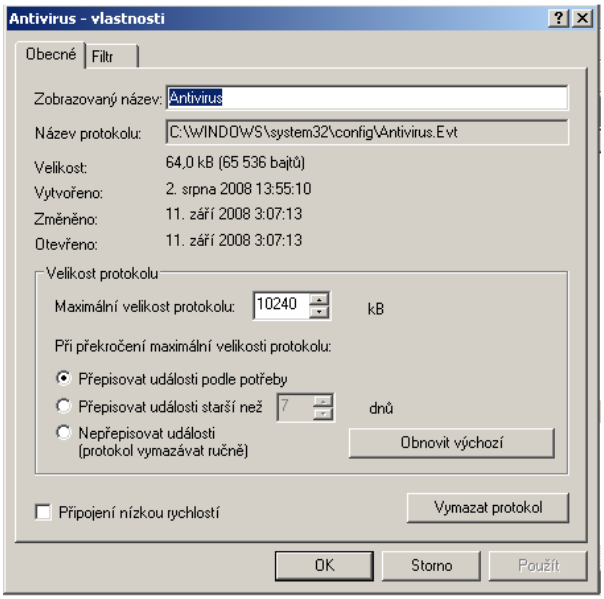
Obdobně důležitá událost v bezpečnostním logu má **EventID 528**. Ukazuje kdy se který uživatel úspěšně přihlásil. Můžeme tak vysledovat zda někdo neoprávněně nepoužívá náš počítač v době naší nepřítomnosti. Pokud někdo Security log vyčistí, je do něj uložena zpráva s **EventID 517**, která identifikuje uživatele který tuto akci provedl a stanici (buď název místního počítače nebo IP adresu vzdáleného).

Ve vlastních programech lze logy jednoduše vytvářet a používat pomocí příslušných funkcí. Například v jazyce C# lze použít třídu EventLog ze jmenného prostoru System.Diagnostics:

```
using System.Diagnostics
private void WriteToEventLog(string message)
{
```

```
 string cs = "Nase aplikace";
 EventLog log = new EventLog();
if (!EventLog.SourceExists(cs))
       EventLog.CreateEventSource(cs, cs);
log.Source = cs;
log.EnableRaisingEvents = true;
log.WriteEntry(message);
}
```

Fyzicky jsou logy v systému uloženy ve formě textových souborů, standardní umístění je %systemroot%\System32\config\, takže například:
C:\WINDOWS\system32\config\AppEvent.Evt
C:\WINDOWS\System32\config\SecEvent.Evt
C:\WINDOWS\system32\config\SysEvent.Evt



Obrázek 83: Vlastnosti souboru logu

Zálohu logů můžeme provést pouhým zkopírováním souboru (není uzamčen).

Jak nainstalovat Group Policy Editor (gpedit.msc) na Windows XP Home Edition

Tento nástroj který nám umožňuje některá pokročilá nastavení zabezpečení počítače bohužel není dostupný na Windows XP Home Edition. Pokud máme k dispozici soubory z Windows XP Professional Edition je možné jej na Home Edition rozchodit. Postup je následující:
Zkopírujeme následující soubory do %systemroot%\system32\

```
appmgmts.dll
appmgr.dll
fde.dll
fdeploy.dll
gpedit.msc
gpedit.dll
gptext.dll
```

Dále pak zkopírovat soubory

```
system.adm
inetres.adm
conf.adm
```

do adresáře %systemroot%\system32\GroupPolicy\Adm\
a nakonec zaregistrovat DLL knihovny z příkazového řádku programem regsvr32.exe:

```
regsvr32 appmgmts.dll
regsvr32 appmgr.dll
regsvr32 fde.dll
regsvr32 fdeploy.dll
regsvr32 gpedit.dll
```

```
regsvr32 gptext.dll
```

Pak je možné používat Group Policy stejně jako na Windows XP Professional Edition (ne všechny volby jsou ale dostupné i na Home) – na příkazovém řádku spustit gpedit.msc.

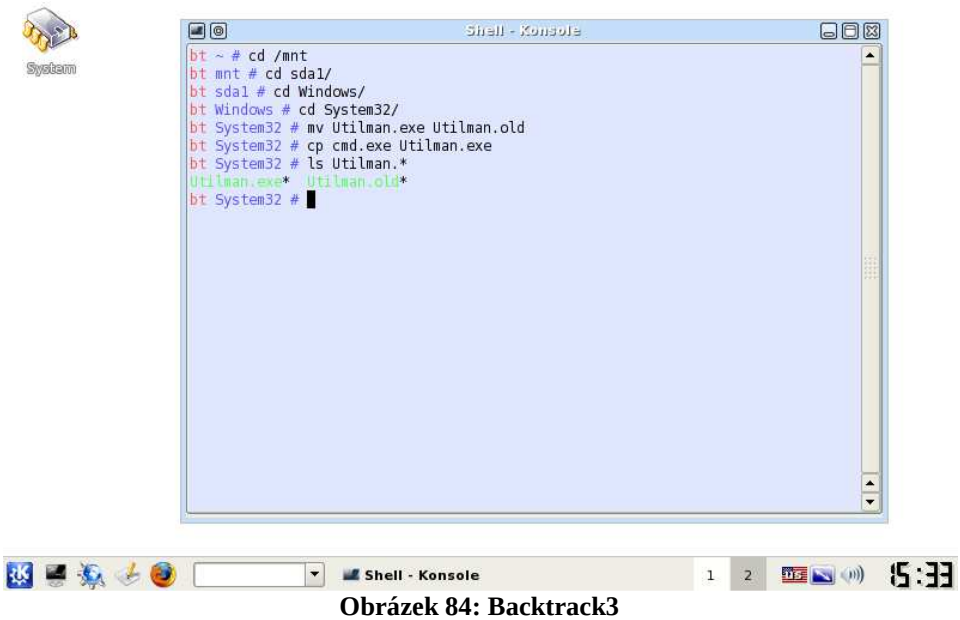
Tip: povolení účtu Administrator ve Windows Vista Home Premium Edition

Na všech verzích Windows Vista je po instalaci účet Administrator „skrytý“ a nelze se na něj přihlásit. Toto se dá změnit v nástroji Místní uživatelé a skupiny (lusrmgr.msc). Windows Vista Home Edition jej neobsahuje a přes nástroj „control userpasswords2“ z příkazové řádky situaci také nespravíme. Pokud nějaký program chceme spustit jako správce, musíme využít kontextové volby přes pravé tlačítky myši a dát „Spustit jako správce“. V případě programů na příkazové řádce to ale nejde. Situace má řešení pomocí příkazu „net“. Postup je následující: spusťte příkazový řádek jako správce (přes ikonu), do spuštěného promptu napište příkaz „net user administrator /active:yes“. Od této chvíle je účet administrátora dostupný a lze se na něj přihlásit (ale nemá nastavené heslo!). Vrátit do výchozího nastavení se lze příkazem „net user administrator /active:no“.

8.3 Windows Vista

Vývoj operačního systému Windows Vista trval několik let, stál stovky milionů dolarů a je propagován jako nejbezpečnější operační systém. Přesto i on má chyby, díky kterým jej dokáže nabourat prakticky každý během pár desítek sekund. Jediné co je potřeba je fyzický přístup k počítači a možnost nabootovat vlastní operační systém. Z CD nebo USB nabootujeme populární bezpečnostní distribuci Backtrack3. Otevřeme konzolu, postupně napíšeme příkazy

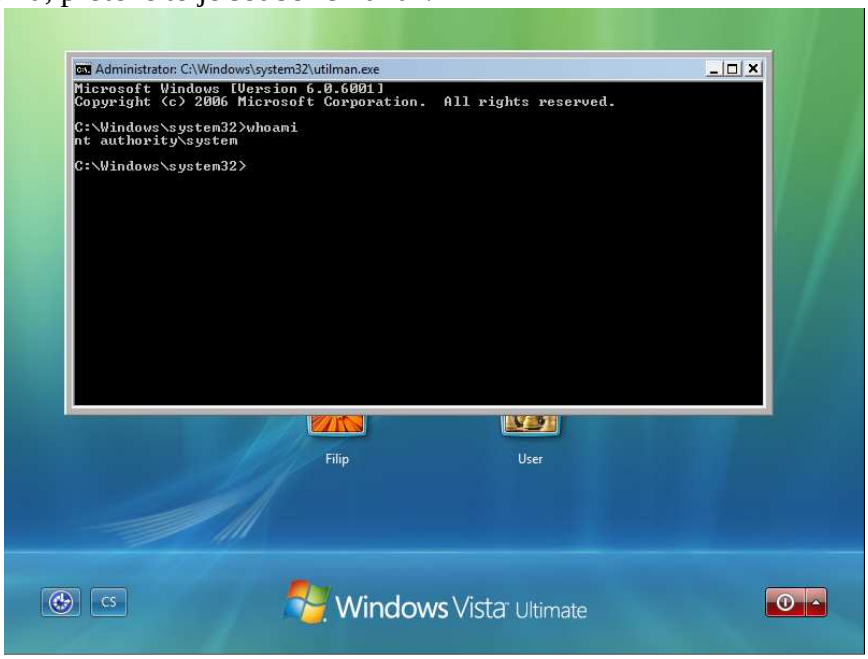
```
cd /mnt
cd sda1/
cd Windows/
cd System32/
mv Utilman.exe Utilman.old
cp cmd.exe Utilman.exe
```



Obrázek 84: Backtrack3

Poté restartujeme počítač, nabootujeme do Windows a na přihlašovací obrazovce stiskneme klávesovou zkratku Windows+U. Místo původního Utilman.exe se spustí soubor, kterým jsme jej nahradili. Navíc jsme přihlášení jako nt autority\system což běžně není možné. Toto si ověříme příkazem „whoami“. Účet SYSTÉM má dokonce má větší práva než účet Administrator. Příkazem „net user uzivatel heslo /add“. Máme totální kontrolu nad systémem, můžeme si přidávat uživatele, instalovat programy, číst hashe.. Ukrást soubory by sice při fyzickém přístupu k hardware bylo

možné i tak, ale takto lze navíc získat přístup do živého systému a v případě že se nám povede nabourat řadič domény Active Directory (viz dále) také do celé sítě.
Program Utilman.exe, který jsme nahradili, nabízí usnadnění uživatelům s handicapem – umožňuje používat virtuální lupu pro zvětšení částí obrazovky, číst text z obrazovky a podobné funkce. V principu ho můžeme nahradit jakýmkoliv programem. Nemůžeme tak nicméně učinit za běhu operačního systému, protože to je soubor chráněn.



Obrázek 85: Napadení Windows Vista přes utilman

Tento postup lze použít také na Windows Server 2008! Kroky jsou stejné jako ve Windows Vista, jen je nutné navíc v adresáři %systemroot%\winsxs\x86_microsoft-windows-utilman_.. přejmenovat Utilman.exe na Utilman.old. V prostředí rozsáhlé Active Directory domény je toto extrémně nebezpečné. Stačí jediný nedůsledně fyzicky zabezpečený server v zapadlé pobočce a lze během pár sekund získat přístup do sebelépe zabezpečené sítě. Pokud útočník neovládá linuxové prostředí, může použít i Windows-based bootovací cd, vytvořené například pomocí Microsoft Deployment Solutions Accelerator.

8.4 Shrnutí

Hesla jsou v operačním systému Windows uchovávána pomocí mechanismu LM hash a NTLM hash. Tyto hodnoty jsou chráněny, nicméně za určitých okolností se k nim lze dostat a s velkou pravděpodobností také rychle prolomit. Pokud má neoprávněná osoba nekontrolovaný fyzický přístup k počítači s operačním systémem Windows, nelze jej považovat za bezpečný.

Úkoly pro cvičení:

- Ve cvičném OS Windows XP ve VMWare si vytvořte deset uživatelských účtů. Jednotlivým účtům vytvořte hesla se vzrůstající komplexicitou. Například prvnímu účtu 123, druhému book, třetímu heslo, čtvrtému Milan, .. poslednímu Ut8d_34R5!B3GGd. Poté ve VMWare nabootujte nástroj ophtcrack, proveďte audit hesel a zkontrolujte do jaké míry komplexnosti byl nástroj schopný odhalit vaše hesla. Pro jednotlivá hesla dále vypočítejte teoretickou dobu vyzkoušení všech možných kombinací za předpokladu že za jednu sekundu jsme schopni otestovat X kombinací.
- Spusťte čistou instalaci Windows XP ve VMWare prostředí s přístupem na Internet. Otestujte za jak dlouho dokážete (kdo první ať je skupina trochu motivována, ať si to ale vyzkouší všichni) surfováním po internetu a otevíráním nebezpečných souborů zavirovat počítač nebo (lépe) jej nakazit některou formou malware natolik že například nepůjde

pracovat s Internet Explorerem (vyskakující okna, neustále se vnucující úvodní stránka apod). Poté zkontrolujte síťový provoz (netstat, wireshark) zjistěte zda nejsou z počítače posílány data ven (např. spamovací bot odesílající desítky mailů za sekundu – tento typ dat jde snadno poznat). Pokud ano, pokuste se zablokovat spojení na firewallu. Pokuste se systém vyléčit.

- Najít kde jsou logy FTP serveru, IIS serveru, sepsat vše (do rozumné délky, cca 1 A4 textu) co se o nich dá zjistit – formáty, volby atd.

Úkoly pro seminární práce:

- Naprogramovat funkci implementující MD5 algoritmus
- Naprogramovat několik iterací rainbow tables (redukční funkci si zvolte libovolnou)
- Mezi jednotlivé posluchače náhodně rozdělit každému 4 nástroje z Top 100 bezpečnostních nástrojů. Úkolem seminární práce je stručně (ale bez vynechání podstatných informací) napsat popis každého z nástrojů (k čemu, jak, syntaxe, příklady, zajímavosti,..) a pokud je dostupný (až na výjimky všechny, byť některé formou zkušebních verzí) vyzkoušet jej a popsat různé scénáře jeho použití.
- Čistá instalace Windows XP. Instalace Wireshark. Odposlechnutí provozu při zapínání počítače. Otevření exploreru se stránkou google.cz. Odposlechnutí provozu. Analýza všech zachycených packetů – do jakého protokolu patří, co má protokol za úkol, co má konkrétní paket za úkol.
- Vygenerovat si vlastní rainbow table pomocí rainbow crack tools, projít celý proces i s použitím

9 Trestní zákon České republiky ve vztahu k malware

Trestní činnost v oblasti výpočetní techniky se nazývá "počítačová kriminalita". 1. ledna 2010 nabyl účinnosti na území České republiky nový trestní zákoník, zákona č. 40/2009 Sb. nahrazující předešlý trestní zákoník č. 140/1961 Sb. Všechna vyjádření této podkapitoly se budou vztahovat k tomuto novému TZ, nebude-li řečeno jinak, a budou směrována k tématu počítačové kriminality.
Do oblasti zabývající se počítačovou bezpečností, osobními údaji či manipulací s informacemi mohou být zařazeny tyto paragrafy:

- § 181 Poškození cizích práv - vztah k phishingu, hoaxu či nevyžádané pošty (spam).
- § 182 Porušení tajemství dopravovaných zpráv - pojednává mimo jiné i o elektronické komunikaci.
- § 230 Neoprávněný přístup k počítačovému systému a nosiči informací - paragraf zabývající se neoprávněným vniknutím do cizího počítačového systému se může aplikovat také na penetraci pomocí malware. Zákon vymezuje odnětí svobody až do výše osmi let, zákazem činnosti nebo propadnutím věci či majetkové hodnoty a to dle závažnosti spáchaného činu.
- § 231 Opatření a přechovávání přístupového zařízení a hesla k počítačovému systému a jiných takových dat - zahrnuje jakékoliv opatření si, uchovávání či distribuci přístupových metod do cizího systému. Vychází z § 182 a § 230
- § 232 Poškození záznamu v počítačovém systému a na nosiči informací a zásah do vybavení počítače z nedbalosti - poukazuje na porušení počítačových systémů z nedbalosti a stanovuje dobu odnětí svobody až na dva roky.

TZ specifikuje i jiné trestní činnosti počítačové kriminality, ty zde ovšem nejsou jmenovány, jelikož přímo nekorespondují s tématem práce.
§ 14 TZ rozděluje trestné činy na přečiny a zločiny - dle jejich závažnosti. Je zřejmé, že úmyslným užití malware a následným vniknutím do cizího systému je zařazeno do kategorie zločinů. Způsobí-li pachatel svým úmyslným jednáním vysokou škodu, může být odsouzen odnětím svobody až do výše osmi let, peněžním, či jiným trestem.
Je důležité si uvědomit, že již samotným přechováváním programu umožňujícího vniknutí do cizího systému je porušován zákon (§ 231) a držitel může být potrestán odnětím svobody v maximální výši jednoho roku, peněžním, či jiným trestem.

9.1 Počítačová kriminalita a historie

"Počítačovou kriminalitu je třeba chápat jako specifickou trestnou činnost, kterou je možné spáchat pouze s pomocí výpočetní techniky, a kde je výpočetní technika předmětem trestného činu nebo pachatelovým nástrojem ke spáchání trestného činu."[15] Malware je tedy nedílnou součástí počítačové kriminality. V roce 2008, dle firmy Symantec Corporation, přesáhl počet různých exemplářů malware hranici jednoho miliónu[16]. Ze zprávy dále vyplývá, že celé dvě třetiny nových hrozeb byly vytvořeny během roku 2007.
V dalším výzkumu firma prezentuje nejvíce rozšířený malware v historii [17].

9.1.1 I love you (2000)

Počítačový červ zaměřený na systémy Windows, vytvořený v jazyce VBScript a šířený pomocí elektronické pošty. K jeho aktivaci je potřeba spuštění ze strany uživatele. Po spuštění

vyhledává emailové adresy aplikace Microsoft Outlook a na nalezené kontakty se rozešle jako příloha, dále přepisuje uživatelské soubory a vytváří záznamy v systémových registrech.
Později bylo vytvořeno i množství modifikací tohoto červa.

9.1.2 Code Red/Code Red II (2001)

Červ zneužívající vzdálené zranitelnosti MS01-033 v systému Microsoft IIS web server. Po aktivaci proběhlo skenování dalších IP adres a také šíření na takto postižené stroje. Červ byl použit např. k útoku typu denial of service, při kterém byly, mimo jiné, postiženy i služby domény whitehouse.gov.

9.1.3 Slammer/Sapphire (2003)

Červ naprogramovaný v assembleru, napadající zranitelnost MS02-039 produktu Microsoft SQL Server 2000 či jiného programu obsahující Microsoft SQL 2000 Desktop Engine. Šíří se výhradně přes síť, na vzdáleném systému využije chyby přetečení zásobníku a usídlí se pouze v operační paměti (na disk se neukládá). Po spuštění náhodně generuje další IP adresy, na které se snaží rozšířit - tím generuje značný internetový provoz.

9.1.4 Sobig (2003)

Jedná se o počítačového červa šířeného pomocí přílohy elektronické pošty. Existuje v šesti variantách, nejvíce se rozšířila varianta Sobig.F. Obsahuje vlastní SMTP engine a není tedy závislý na konkrétním poštovním klientovi. Červ musí být spuštěn uživatelem, poté se nahraje na disk a začíná prohledávat místní soubory a vyhledávat v nich řetězce podobné emailovým adresám. Mezi další vlastnosti patří šíření pomocí síťového sdílení, vytvoření zadních vrátek systému, či stahování instrukcí k dalšímu šíření. Tento malware byl navrhnut k ilegálním komerčním účelům - vytvoření sítě botnet a následného šíření spamu. Mydoom (2004)
Červ cílený na platformu Microsoft Windows vytváří po spuštění zadní vrátka do systému a snaží se šířit sám sebe na další stroje. Jeho varianty Mydoom.A a Mydoom.B jsou šířeny přes elektronickou poštu a P2P sítě- v příloze pošty je zasílán v různých formátech (s různými příponami). Je známa také varianta Mydoom.C, která se snaží využít již existujících zadních vrátek systému, vytvořených předchozí variantou tohoto červa a provést aktualizaci. Mydoom.B brání v přístupu na webové stránky firmy Microsoft a známých antivirových společností.

Název	Počet infekcí	Způsobená škoda
I love you	500 000	15 000 000 000$
Code Red	1 000 000	2 600 000 000$
Slammer	200 000	1 200 000 000$
Sobig.f	2 000 000	37 100 000 000$
Mydoom	2 000 000	38 000 000 000$

Tento výčet zdaleka není kompletní, ale je dobrou ukázkou rozmanitostí různých malware a škod jimi napáchaných.

9.2 Motiv útočníka

Ptát se "Proč lidé vytváří počítačové viry?" je stejné jako otázka "Proč lidé páchají zločiny?"[18]. V širším slova smyslu nemusíme vždy najít odpověď, zaměříme-li se však na jednotlivce či skupiny, můžeme ze známých případů odvodit záměry, se kterými útočníci malware tvoří.
Většinu úmyslů zahrnuje tento seznam[19]:

- Krádež citlivých informací - útočník se snaží získat údaje o platebních kartách, heslech, osobách, společnostech či ukrást citlivá data v podobě datových souborů. Ukradené informace může použít buď pro své vlastní potřeby či prodat jiné osobě, popřípadě nabídnout na černém trhu.
- Generování zisku - tvůrce může využít napadené počítače k rozesílání nevyžádané pošty či pronajímat síť napadených počítačů. Ty pak mohou být nájemcem užity rovněž k rozesílání pošty, dále k provádění útoků, především typu denial of service a crackování hesel hrubou silou.
- Převzetí kontroly nad zařízením - útočník může chtít pouze získat přístup k určitému zařízení. Buď z důvodů osobních (pomsta, vydírání, atd.), či na zakázku.
- Poškození - cílem může být také pouhé poškození systému, sítě, či poškození společnosti, na niž je útok směřován.
- Důkaz schopností - předvést své schopnosti ostatním může být dalším z motivů útočníka. Malware stvořený pro tento účel nemusí nést nebezpečný náklad, ale může obětem zobrazovat nevyžádaná sdělení (hlášení systému, vtipné animace či jiný kreativní obsah).

9.3 Definice a struktura malware

"Malware je souhrnné označení pro potenciálně nebezpečný či přímo škodlivý software. Pochází z anglického MALicious softWARE. Zahrnuje nepřeberné množství rizikových aplikací, jako jsou viry, červi (worms), trojští koně, spyware, rootkity apod."[20]
Malware nemá vždy jednoznačnou strukturu, protože ta je závislá na typu systému, pod kterým bude malware provozován. Obecně se však jedná o proveditelný kód nebo taková data, která způsobí po svém načtení (nadřazeným systémem) útočníkem zamýšlenou akci. Existují také typy malware, které využívají chyb autorů cílového systému ke své vlastní činnosti.
Lze tedy uvést jen obecnou strukturu. Ne všechen malware obsahuje tyto části, jiný typ malware může naopak obsahovat části zde neuvedené:

- Hlavička - některé typy souborů potřebují ke svému spuštění hlavičku, obsahující metainformace potřebné k dalším úkonům. Příkladem může být PE hlavička proveditelných souborů Microsoft Windows.
- Tělo - vlastní tělo počítačového červa, viru či trojského koně. Obsahuje samotný kód, který má malware provádět. Dle typu může být v binární či textové podobě.
- Náklad (anglicky "payload") - útočníkem definovaná data, která budou nahrána do systému po úspěšném spuštění malware.
- Dekryptor - v rámci maskování a ochrany před antiviry může být tělo viru zašifrované. Dekryptor se aktivuje po spuštění viru a většinou v reálném čase provádí dešifrování jeho těla. V případě polymorfních virů také může smysluplně zaměňovat pořadí prováděných instrukcí.

9.4 Rozdělení malware

Malware se dělí do různých kategorií - dle účelu, způsobu šíření a penetrace, dle cílového hostitele a dalších kritérií.
Dělení dle účelu:

- Nakažlivý malware - počítačoví červi a viry
- Utajení - trojští koně
- Sledování - spyware
- Reklamní aplikace - adware

Dále jmenované typy nákaz jsou podmnožinou malware[12]:

9.4.1 Počítačový červ

Autonomní software, jenž vytváří kopie sama sebe a ty dále šíří především počítačovou sítí formou síťových paketů (někdy se také pojmem počítačového červa označují i takový malware, který se šíří elektronickou poštou). K infekci dochází ve většině případů přes zranitelnost cílového počítačového systému. Po úspěšném napadení vykoná nejen naprogramované příkazy, ale především se zkusí dále šířit.

9.4.2 Počítačový virus

Název je odvozen od virů biologických, jelikož je svým chováním připomínají. Aby byl virus proveditelný, musí být přeložen do formátu, ve kterém jej bude cílový operační systém schopen spustit. Virus může být spuštěn i jinou entitou, než operačním systémem.
Podle místa působení dělíme viry dále do kategorií:

- Souborový virus - bývá distribuován ve formě proveditelného kódu. Většinou je autorem předem definována množina operačních systémů, pod kterou bude virus schopen vykonávat svou činnost. Souborový virus pak může své kopie vytvářet, přepisovat jiné soubory (přepisující viry), či se k jiným spustitelným souborům připojovat (parazitické viry).
- Boot viry - napadají master boot záznamy a zaváděcí sektory vyměnitelných médií. Virus bude spuštěn ihned, jakmile mu BIOS předá zavádění systému. Nahraje se do paměti (a stane se paměťově rezidentním), obsadí adresy systémových služeb a samozřejmě zavede původní systém.
- Skriptové viry - viry vytvořené pomocí skriptovacího jazyka daného systému. Je možné využít pouze ty možnosti, které jsou přes skriptovací jazyk přístupné. Bývají uložené jako skript uvnitř dávkového souboru a může být velmi jednoduché tyto viry napsat.
- Makroviry - určité programy umožňují uživateli automatizaci a zjednodušení práce pomocí tzv. "maker". Nejčastěji tuto funkci nabízí kancelářské aplikace. Jsou tedy podobné skriptovým virům, ale zde jej obsluhuje jiný software než operační systém. Makrovir využívá jen funkce nadřazeného programu a cílového makrojazyka.

Viry mohou být uloženy i jiným způsobem - např. jako datové struktury, musí jej ovšem aplikace, pro kterou jsou určeny, spustit, jinak se stávají bezcennými.

9.4.3 Trojští koně

Jsou připojeni jako nežádoucí kód k jinému software, který je pro uživatele žádoucí. Nejsou schopni sebe-replikace a musí být tedy staženi a spuštěni uživatelem. Bez vědomí uživatele ovšem vykonává svou předem definovanou činnost. Nejčastěji vytváří zadní vrátka do systému.

9.4.4 Spyware

Malware zaměřený především ke sledování a získávání dat z cílového systému. Může být naprogramován k reprodukci, ale nemusí. Jeho prioritou je odesílání získaných informací a maximální možnosti utajení.

9.4.5 Adware

Program podporující reklamní či propagační činnost, který je vložen do jiného software produktu. Většinou se jedná o legální kód, který neprovádí žádnou škodlivou činnost. Adware se vyskytuje většinou ve volně dostupných aplikacích a cestou, kdy uživateli zobrazuje reklamu, financuje svůj další rozvoj.
Stává se, že někdy je obtížné zařadit škodlivý program do přesné kategorie. Proto je přirozené, že specifický malware může obsahovat prvky různých kategorií. Některé publikace také uvádí další rozdělení, pro tuto práci jsou ovšem informace dostačující. Kategorizace činností, které malware může provádět, bude popsána později, stejně tak možnosti penetrace, či užití metod sociálního inženýrství.

9.5 Důležitá příprava

Před tvorbou malware si autor zpravidla musí ujasnit záměry a cíle své práce.

- morální a trestní odpovědnost
- časový harmonogram a životnost projektu
- požadované výsledky
- cílová skupina systémů a uživatelů
- možnosti penetrace
- šíření malware
- náklad / vykonávané akce
- utajení
- jiné specifické úkony

Dále pak tvůrce zhodnotí své schopnosti a znalosti, případně potřebné informace nastuduje. Logicky vyplývá, že k naprogramování účinného malware potřebuje autor především poznatky o cílovém systému a dovednosti v algoritmizaci. Znalostem a dovednostem pak přirozeně musí přizpůsobit i svůj výsledný produkt.

9.6 Sběr informací

Vytváření malware se v různých bodech shoduje s hackerskými postupy. Útočník také potřebuje dobře znát svůj cíl a získat o něm, pokud možno, co nejvíce informací. "Prvním krokem hackerského procesu je získávání informací o cíli. Získané informace se nazývají otisky prstů. V době internetu jsou kousky informací dostupné z různých zdrojů"[5].

Záleží ovšem, na jaký cíl se útočník chce zaměřit:

- Specifický cíl - subjektem může být jednotlivec, skupina, společnost či organizace. Tvůrce malware se tedy zaměří na svůj cíl, o kterém se bude snažit získat co nejvíce informací. Především pak o síťové infrastruktuře subjektu, konkrétních systémech, jejich odlišnostech a konfiguracích.
- Obecný cíl - spíše než na konkrétní subjekt se tvůrce zaměří na určitou technologii a systémy. V zásadě útočník neřeší otázku "Kdo systémy užívá?", ale řeší množství uživatelů technologie. To je důležité v situaci, kdy cílem útočníka je napadnout, pokud možno, co nejvíce strojů a vytvořit z nich ilegální síť. Potřebné informace k útoku na obecný cíl: možnosti vykonávání kódu uvnitř napadeného systému, bezpečnostní slabiny, druhy komunikace a jiné.

Další části kapitoly se zaměřují především na získávání informací o konkrétním cíli. Tvůrce malware si musí ověřit, jakým způsobem je jemu cílový systém dostupný:

- Fyzická dosažitelnost - útočník má přímý přístup k zařízení. V tomto případě bývá napadení zpravidla nejjednodušší. Útočník může sám na stroji spustit svůj malware a nechat vykonávat jeho činnost.
- Vzdálená dosažitelnost - výhodou, a současnou slabinou, informačních technologií je možnost vzdáleného přístupu k nim skrze síť. Přístup skrze síť nemusí automaticky znamenat, že je možné se k cíli dostat přímo či nepřímo přes internet - cíl může být přístupný pouze z autonomního systému, ke kterému je připojen. Zda-li je možné se k němu dostat i z internetu záleží na vnitřní infrastruktuře autonomního systému a také na nastaveném směrování, překladu adres a vnitřním zabezpečení.
- Žádná dosažitelnost - existují stanice, které jsou z různých důvodů od sítě úplně odpojeny. To ovšem neznamená, že není možné je jistým způsobem nakazit. Útočník může svůj malware poskytnou na datovém nosiči, který bude jinou osobou otevřen a následně spuštěn. Tato záležitost spadá spíše do oblasti sociálního inženýrství, která bude probírána později. Problém ovšem nastává v případě pokusu o získání dat z takto nepřístupného stroje. Útočník je v této situaci opět závislý na zásahu třetí strany, která má možnost přístupu.

Následující podkapitoly se budou věnovat získávání informací vzdáleně, prostřednictvím sítě. Je potřeba uvést rozdělení, kdy je cíl připojen k vnitřní síti stejně jako útočník, či je útočník připojen přes síť vnější. Druhá z možností zahrnuje také přístup skrze síť internet. V textu je předpokládáno, že veškeré síťové prvky komunikují pomocí modelu TCP/IP.

9.6.1 Informace o vzdáleném systému skrze vnitřní síť

Uvažujme, že cíl je přímo dosažitelný skrze síť (není tedy odstíněn síťovou infrastrukturou). Útočník zajisté bude chtít zjistit co možná nejvíce informací, před penetrací to ovšem bude možné jen v určité míře. Množství vzdáleně zjistitelných informací závisí na konkrétních službách běžících na cílovém stroji. Ne všechny informace budou také k penetraci použitelné.
Nejdříve je zapotřebí znát cílovou IP adresu. IP adresa je používána při komunikaci na 3. vrstvě referenčního modelu IOS/OSI. Bude-li útok probíhat z vnitřní strany sítě, je samozřejmě podstatné nalézt adresu vnitřní a ne vnější. Nezná-li ji útočník, může využít různých metod, jakými si ji opatřit. Zaslat DNS dotaz, odeslat ARP dotaz, odchytávat komunikaci, vyčíst ji z logovacích souborů uložených při dřívější komunikaci a to buď z lokálního systému, nebo z jiného stroje, se kterým cíl komunikoval. Možností je více, především také využitím sociálního inženýrství (viz dále) - příkladem může být i fiktivní webová aplikace, na kterou je oběť dovedena phishingovými triky.

Následuje skenování portů. "Skenování portů je proces, kdy se snažíme zjistit, jaké síťové služby běží na dané IP adrese"[21]. Většina síťových služeb má předdefinován port, na kterém se v základní konfiguraci spouští. Mnoho administrátorů nechává porty nezměněné, tím ulehčí útočníkovi identifikaci služby. Existují databáze portů[22], kde může každý dohledat aplikaci na tomto portu běžící (v základní konfiguraci). Útočník si tak udělá představu o cílovém systému. Především také o operačním systému, jelikož ne všechny služby jsou podporovány každým systémem. Pachatel se dále může zkoušet připojit na různé služby a hledat další informace skrze ně. Častým případem může být připojení na službu webového serveru. Ze strany provozovatele bývá většinou zapnutý záznam (log), který ukládá všechny zaslané žádosti na tuto službu - datum a čas, zdrojové IP adresy a požadavek samotný.
Útočník má v místní síti výhodu, jelikož se může pokusit provést útok na síťovou infrastrukturu, DNS či jinou službu. Nejvíce zajímavé bude pro pachatele odposlouchávání cizí komunikace - software zachytává rámce/pakety určené pro oběť a ukládá jejich obsah na útočníkův stroj. Odposlech je možný jak v rámci drátové, tak bezdrátové sítě. Bezdrátová síť má v tomto ohledu, z pohledu zabezpečení, nevýhodu, jelikož signál je šířen prostorem a je tedy lehčí jej odchytit, na druhou stranu dnešní přístupové body zvládají signál bezpečně zašifrovat. Odposlech je v aktuálním případě pasivním typem útoku, kdy provoz není měněn.
Aby mohl útočník odposlouchávat na přepínaných sítích, musí zajistit, aby se požadovaný provoz dostal na jeho linku. Nejjednodušší, ale často nejméně proveditelné, je připojit zařízení přímo na páteřní síť, či pomocí habu do požadované části síťové topologie. Některé síťové prvky umožňují v rámci ladícího módu nastavit Switched Port Analyzer a nechat na nastavený port zasílat veškerý provoz. Útočník ale musí získat přístup ke konfiguraci daného prvku.
Jiné možnosti spočívají v provedení aktivního odposlechu - musí být proveden útok na síť, který způsobí odklonění provozu. Nejpoužívanější techniky se nazývají[3][5] ARP cache poisoning, DNS poisoning, MAC flooding, či jiné.
Práce se jimi nebude podrobněji zabývat, jelikož se tematicky jedná o postupy používané spíše v hackerské praxi.

9.6.2 Informace o vzdáleném systému skrze vnější síť

Zpravidla je vždy přístup z vnější sítě více zabezpečený, než přístup ze sítě vnitřní. Cílový systém dokonce nemusí být z vnější sítě přístupný - nachází se za směrovačem, který provádí překlad adres (Network Address Translation, či Network Masquerading) a je chráněn firewallem. Není-li přímo na hraničním směrovači nastaveno směrování do vnitřní sítě, nebude možné se na vnitřní server z vnější sítě připojit.
Chce-li se útočník dostat do vnitřní sítě, musí nejdříve získat kontrolu nad hraničním směrovačem. Možností je několik. V minulosti se již objevily zranitelnosti, pomocí kterých je možné ovládnout směrovač [23] (včetně chyby krvácejícího srdce) - u některých je ovšem zapotřebí provést útok z vnitřní sítě. Útočník může vyzkoušet připojení do vnitřní infrastruktury pomocí slabě zabezpečené bezdrátové sítě. Další možností je útok hrubou silou na vzdálenou správu zařízení, VPN koncentrátor či pokus o napadení jiné služby. Je-li ve firewallu povoleno připojení na jiný systém uvnitř sítě, může pachatel zaútočit na něj a z napadeného systému provést další útok, tentokrát na vnitřní cíl.
Další útoky pak již počítají s akcí ze strany oběti, která vytvoří spojení. Poměrně novým typem útoku je Drive-By Pharming[13], který po navštívení webové stránky uživatelem spustí JavaScript, jež se snaží připojit k administraci výchozí brány (za pomoci výchozího hesla) a změnit adresy DNS serverů. Chce-li pachatel zaútočit na překlad adres, může využít techniku NAT Pinning[24], kdy oběť navštíví kompromitovanou webovou stránku, ta se snaží otevřít spojení, které se pro směrovač jeví jako spojení jiného než HTTP(S) protokolu a pro které je potřeba otevřít nový lokální port. Směrovač se proto v dobré vůli pokusí využít technologii NAT traversal, pro kterou útočník specifikuje požadovaný port, ten je přesměrován a pachatel jej

může využít. Technikou Drive-By Pharming a NAT Pinning lze napadnout pouze směrovače na tyto útoky náchylné.
Nejčastěji dochází k napadení oběti za firewallem opět přes kompromitovanou webovou stránku, která využije zranitelnosti internetového prohlížeče či aktivního modulu. Ke zjištění informací o vzdáleném stroji, který je schován za firewallem, ovšem není potřeba ani nelegální činnosti. Programátor vytvoří webovou stránku, která se po otevření pokusí z cílového spojení, zaslaného HTTP požadavku a webového prohlížeče získat potřebné informace. Ze spojení přečte především IP adresu, z HTTP hlavičky pak mnohem více údajů - užitý operační systém a prohlížeč, včetně jeho verze (User-Agent), předchozí zobrazenou stránku (Referer), jazykovou mutaci (Accept-Language) či jiné. Webový prohlížeč může poskytnout cookies třetích stran, informace skrze JavaScript (počet aktivních modulů, rozměr okna, počet instalovaných písem atd.)[25] a zásuvné moduly. Z těchto otisků si může autor udělat představu o cílovém systému a stává se pro útočníka mnohem lépe identifikovatelným[26].
Zbývá vyřešit, jakým způsobem přiměje pachatel oběť zavítat na takovýto web. Přesměrování provozu vzdáleného cíle je v prostředí internetu velmi problematické, proto bývá častým scénářem použití metod sociálního inženýrství.

9.6.3 Sociální inženýrství

"Sociální inženýrství užívá manipulaci, vliv a podvod k přinucení osoby, zasvěcené uvnitř cílené organizace, splnit požadavek, který obvykle zahrnuje zveřejnění informace nebo vykonání akce, která napomůže útočníkovi. Může se jednat o prostý telefonát až po více komplexní situaci - přinucení oběti navštívit webovou stránku, která využije technické zranitelnosti a dovolí hackerovi převzít kontrolu nad počítačem.
Druh tohoto útoku se často označuje termínem "netechnický hacking". V oblasti informačních technologií se metody proslavili v 80. letech 20. století, kdy se o jejich popularizaci zasloužila mediální kauza kolem bývalého amerického hackera a sociálního inženýra Kevina Mitnicka. V dnešní době se jedná o jednu z nejrozšířenějších metod, jak z oběti získat požadované informace, či ji donutit udělat útočníkem zamýšlenou akci.
V roce 2011 zveřejnila firma Check Point® Software Technologies Ltd hlášení[27] z výzkumu "The Risk of Social Engineering on Information Security", ve kterém uvádí, že 48% z dotázaných společností se staly terčem útoku sociálního inženýrství 25 krát a více za uplynulé dva roky. Jeden úspěšný útok způsobil firmě finanční ztrátu 25 000$ až 100 000$, i více. Z hlášení vyplývá, že útoky byly zaměřené (seřazeno sestupně od nejvíce cíleného) na nové zaměstnance, dodavatele, výkonné ředitele, oddělení lidských zdrojů, vedení podniku a nakonec na informační oddělení.
Útočníci se snaží metodami sociálního inženýrství využít znalosti psychologie a faktory, jež ovlivňují chování a úsudek oběti, mezi které se řadí: důvěra, sympatie, nekonfliktnost, respekt, stres, soucit či jiné. Úspěšnost útoku závisí především na aktuální situaci, ve které se oběť nachází, na její povaze, pocitech, dále informovanosti, technických znalostech a různých jiných faktorech. Ve většině případů jsou rozhodující také schopnosti útočníka, zda dokáže v oběti vyvolat výše zmíněný pocit, či navodit požadovanou atmosféru. Dalším ovlivňujícím faktorem je vztah oběti k útočníkovi a délka známosti, či množství předchozí komunikace.
Sociální inženýr může využít různé druhy komunikace: osobní styk, telefonický hovor, zpráva, webová stránka, osobu třetí strany, atd. Ke každé z možností se může oběť stavět s jinou důvěrou a pozorností. Osobnímu styku či telefonátu bude oběť věnovat větší pozornost, než například elektronické zprávě. Elektronickou zprávou na druhou stranu může útočník oslovit velké množství adresátů za krátký čas. Sociální inženýr tedy musí vždy předem plánovat a vyhodnocovat své techniky.
Sociální inženýrství zavádí terminologii pro specifické metody a úkony:

- Phishing[8] - odeslání falšovaného e-mailu příjemci, který klamavým způsobem napodobuje legální instituci s úmyslem vyzvědět od příjemce důvěrné informace. Příjemce je většinou ve zprávě nabádán ke vstupu na webovou stránku přes přiložený odkaz. Ve scénáři phishingu je tato stránka vytvořena útočníkem a vzhledově napodobuje webové stránky instituce, za kterou se odesílatel vydává. Obsah zprávy či webové stránky vybízí oběť k zadání osobních údajů, přihlašovacích údajů k určité službě nebo k zadání čísla platební karty. Nerozpozná-li oběť, že se jedná o plagiát a vyplní-li důvěrné informace, jsou podvodnou stránkou uloženy, či odeslány útočníkovi. Tvůrce stránky může definovat, jaký další obsah bude oběti zobrazen. Může se jednat o chybovou zprávu s následným přesměrováním na legitimní webové stránky společnosti, za kterou se ve zprávě vydává.
- Spam - nevyžádané poštovní sdělení šířené elektronickou zprávou. Většinou jsou zprávy zasílány hromadně, kdy cílí na mnoho příjemců. Adresy obětí jsou ve větším měřítku získávány různým způsobem - ukradeny z databází, automaticky vyhledány na internetu pomocí robotů, zaznamenávány přes podvodné stránky a další. Obsah zprávy pak obsahuje reklamní sdělení, phishingové sdělení a může také nést v příloze malware.
- Vishing - metoda, při níž je útok veden telefonickým hovorem namísto elektronickou poštou. Útočník se vydává za jinou osobu, nejčastěji zaměstnance legitimní společnosti, a snaží se z oběti vyzvědět potřebné informace.
- Pretexting - metody utváření vymyšleného scénáře útočníkem a jeho následné sdílení s obětí za účelem přinucení oběti vyzrazení informací či provedení akce. Falšována mohou být fakta, situace a především identita útočníka. Útočník často zakládá části informací na pravdě, aby navodil domněnku legitimnosti akce.
- Baiting - obrazně se dá přirovnat k útoku trojského koně v reálném světě. Pachatel zanechá na místě přístupném oběti zpravidla datový nosič či dokument označený lákavě vyhlížejícím popisem. Útočník doufá, že dříve či později se oběť pokusí spustit vnitřní obsah.

Z hlediska přístupu útočníka k oběti mohou nastat tyto scénáře:

- Přímý přístup - útočník se ptá přímým způsobem oběti na požadované informace.
- Důležitá osoba - útočník se identifikuje jako osoba z vedení firmy a většinou předstírá, že se nachází v situaci, kdy potřebuje rychle vyřešit určitý technický problém. Zdůrazní důležitost požadovaného činu či informací, na které se oběti zeptá. Pachatel v této situaci spoléhá na fakt, že dotyčná osoba uvěří identifikaci, bude jej respektovat jako nadřízeného a ráda pomůže.
- Bezmocná osoba - pachatel se identifikuje jako nový zaměstnanec či zaměstnanec stávající, který řeší problém se systémem. Cílem útoku většinou bývá technická podpora či správce systému. Útočník často předstírá jistou neznalost a doptává se na informace. V údernější variantě scénáře může předstírat, že řeší zapomenuté heslo či první pokus o přihlášení.
- Osoba technické podpory - pachatel se oběti snaží prokázat, že patří k technické podpoře nebo je on sám správcem systému. Uvěří-li oběť této lži, zpravidla pak sdělí požadované informace nebo vyplní útočníkem zadaný úkol.
- Reverzní sociální inženýrství - technika, kdy se útočník pokusí zaranžovat události takovým způsobem, aby jej oběť sama kontaktovala s prosbou o pomoc. Obvykle je útok proveden ve třech krocích:
    - Sabotáž - útočník zinscenuje nějaký problém a snaží se oběti na něj poukázat. Problém může být skutečný nebo fiktivní.

- Inzerce - pachatel nabídne technické řešení odstraňující problém či odbornou konzultaci a vyčkává, zda-li oběť zareaguje na jeho nabídku.
- Pomoc - pachatel problém odstraní, nebo prohlásí za odstraněný. Při vykonávání práce získává útočník informace od oběti a často si také vyžádá přístup do systému.

9.7 Cílový systém

V této fázi má pachatel potřebné, nebo alespoň směrodatné, informace o systému a službách běžících na cílovém stroji. Ze zjištěných informací si útočník udělá obraz o možnostech, které vzdálený systém nabízí. "Jedním z nejdůležitějších kroků pro pochopení počítačových virů je především dobré pochopení prostředí, ve kterém operují."[4] Každé prostředí nabízí jisté zdroje, komunikační prostředky a definuje závislosti, především pak formát proveditelného kódu. Programátor může využít ty možnosti, které mu prostředí poskytuje, ale současně je nucen dodržovat formáty a pravidla prostředím definovaná.
Existují mnohé závislosti, které ovlivňují vykonávání a funkce malware[4]:

- Závislost na počítačové architektuře
- Závislost na procesoru či jiném hardware
- Závislost na operačním systému a jeho verzi
- Závislost na souborovém systému
- Závislost na jiném homogenním subsystémů či programu
- Závislost na zranitelnosti
- Závislost na síťovém připojení

Nepsaným pravidlem v oblasti vývoje software je jeho dělení do vrstev a abstrakce. Tím se řeší problémy s přenositelností kódu. I běhová prostředí se dělí dle vrstev, na kterých pracují:

9.7.1 Strojový kód

Ze software hlediska se jedná o nejnižší vrstvu, ve které může být program napsán - počítač dokáže zpracovávat jen určitý soubor instrukcí, které jsou definovány jeho procesorem. Instrukce jsou zapsány číselnými kódy a ovlivňují chování procesoru. Chce-li útočník vytvořit škodlivý kód pro tuto vrstvu, bude moci využít všechny možnosti procesoru, ale jeho kód bude limitován jen na kompatibilní procesory. Nebude-li řečeno jinak, bude se práce zabývat procesory architektury x86.

9.7.2 Operační systém

Existují různé operační systémy pro různé potřeby a pro různá zařízení. Práce bude směřována především na nejvíce používané OS - Microsoft Windows, MAC OS a Linux. OS obecně umožňují komfortnějšího ovládání počítače a práci s hardware. Odstraňují také přímé závislosti uživatelských programů na konkrétním hardware - vytváří abstraktní vrstvu. Ve skutečnosti je OS ještě více rozvrstven.
Mezi nejdůležitější funkce systému patří: řízení přístupu k hardware, správa procesů a užití multitaskingu, správa paměti, práce se souborovými systémy, práce se sítí a jiné. V neposlední řadě zajišťuje také běh uživatelských programů.
Podaří-li se útočníkovi zavést svůj malware do systému, může využít všech možností, které mu OS nabízí. Často malware využívá API, které umožní využívat prostředky a funkce systému. Na druhou stranu každý z výše jmenovaných OS obsahuje (v aktuální verzi) bezpečnostní mechanismy, které limitují přístup aplikací přes rozhraní systému. Jedná se především o práva

procesů. Každý nový proces se spouští pod určeným uživatelem, který má definovaná svá práva a přístup k souborům. Proces může při spuštění, i za běhu, požádat o jiná práva. Tato akce ale musí být potvrzena, a nebo musí být zadáno heslo požadovaného uživatele.
Při skenování služeb (viz kapitola "sběr informací") se pachateli podařilo získat informace o běžících službách. Jejich chod je zajišťován právě operačním systémem.

9.7.3 Běhová prostředí nad operačním systémem

Další vyšší vrstvou jsou běhová prostředí nad operačním systémem. Užívají určitý stupeň virtualizace a poskytují tak své vlastní funkce programům, které jimi budou spuštěny. Široká vrstva software využívá právě tuto vrstvu, včetně skriptovacích jazyků - dá se říci, že i webový prohlížeč interpretující JavaScript se řadí do této vrstvy, stejně tak kancelářská aplikace, jež umožňuje běh maker.
Tuto vrstvu vytváří i další produkty jako WINE, který zpracovává volání programů určených pro jiný systém a poskytuje vlastní knihovny. Dále pak interpreti bytekódu, kteří překládají bytekód na nativní - Common Language Runtime (.NET aplikace), Java Runtime Environment (Java aplikace) a další.
Je patrné, že různé závislosti způsobují nekompatibilitu mezi různými systémy. Na druhou stranu stačí, když malware splňuje požadavky jen toho systému, pod kterým je zamýšlen jeho chod. Příkladem buď virus implementovaný pro prostředí Oracle Java SE. Jelikož je možné prostředí Java Runtime Environment (JRE) provozovat nad operačními systémy (32 bitovými i 64 bitovými) Microsoft Windows, Apple Mac OS, Linux či jinými, bude možné spustit malware pod těmito systémy a nevzniká na této vrstvě přímá závislost na operační systém či architektuře. Vzniká ovšem závislost na subsystém JRE, který užívá a poskytuje zdroje nadřazeného systému. Pod různými druhy JRE tedy nemusí být vždy malware provozuschopný či přesně vykonávat své funkce. Dále vzniká závislost nepřímá, kdy funkce JRE jsou závislé na možnostech operačního systému.
Je potřeba zmínit, že vrstvení není vždy ve všech ohledech přesně definováno a abstrakce nemusí být ve všech částech systému úplná. Systémem může být například dovoleno přepsání jiného paměťového místa, než určeného pro konkrétní proces, dále i přímí přístup k instrukcím procesoru a jiné. Je tedy vidět, že různé části vrstev se mohou prolínat.

9.8 Tvorba malware

Nyní se následné kroky pachatele rozdělí na dva různé scénáře:

1. Útočník se pokusí o nalezení vzdáleného místa náchylného na exploit. Podaří-li se, bude v návrhu a při programování již s touto možností počítat a přizpůsobí malware a jeho šíření tak, aby využíval nalezené zranitelnosti. Jinými slovy - nejdříve dochází k penetraci a následnému vytvoření malware.
2. Obrácený postup. Útočník se již předem rozhodl, že nejdříve vytvoří malware pro cílový systém a následně se jej bude pokoušet přenést a spustit. Tento postup je aplikován také v případě, kdy útočník nenalezne žádnou vzdáleně zneužitelnou slabinu a bude chtít penetraci provést jiným způsobem, než exploitem.

9.8.1 Programování

Před začátkem samotného programování si autor musí zvolit, pro jaký běhový systém bude aplikaci psát. Je výhodné mít také definovány akce, které po malware autor požaduje. Definované požadavky na malware mu ulehčí určit, jaký běhový systém bude nejvhodnější. Příklad: pachatel bude chtít využít známé zranitelnosti služby a implementovat ji do

počítačového červa, který bude měnit registry systému a dále se šířit - proto je jasné, že nevyužije makro kancelářské aplikace, která úkol nezvládne.
Možná efektivita výsledného kódu bývá často přímo úměrná znalostem útočníka - jinými slovy, "čím více útočník cílové prostředí zná a dokáže využít nabízené možnosti, tím je schopen vytvořit sofistikovanější hrozbu". Je tedy nezbytné, aby si prostředí předem nastudoval. Cenným zdrojem informací jsou příručky k danému prostředí a publikace zabývající se zvoleným prostředím. Útočník si zpravidla dané prostředí nainstaluje na vlastní (i virtuální) stroj a současně s implementací svůj výtvor testuje.
Je běžné, že v průběhu studia, programování či testování nalezne útočník nová řešení, či jen změní názor na určité principy fungování malware. Závisí pak na jeho vůli, zda nové záměry aplikuje do produktu a nahradí tak část své práce.
K tomuto scénáři může dojít i v pozdější fázi, kdy je malware již aktivní na cílových stanicích. Vytvořil-li si autor přístup k napadeným stanicím pomocí svého malware, či existuje-li ještě cesta, jak stanici znova infikovat, může útočník provést aktualizaci své předchozí verze.
Otázka ohledně volby programovacího jazyka není příliš složitá - je ovšem potřeba si uvědomit, že programovací jazyk je pouhý nástroj, se kterým autor pracuje. Více než výborná znalost jazyka je potřeba ovládat umění algoritmizace a programátorských postupů. Mezi potřebné postupy se řadí práce se soubory, s databázemi, se sítí a sokety, vícevláknové programování a další. Různé jazyky mohou přistupovat k těmto technikám trochu jiným způsobem, ale princip je prakticky stejný. Požadovaný programovací jazyk je v konečném důsledku závislý na kompilátoru či interpretačním software. Příkladem je Microsoft .NET Framework, pro který je možno využít spousty jazyků[28] (Visual C# .NET, Visual C++ .NET, Visual Basic .NET, JScript, Python pro Microsoft .NET a další), kompilátor pak zvolený jazyk převede na jednotný bytekód.
Časté řešení ovšem bývá v užití jazyka assembler pro programování malware na nejnižší vrstvě, v rovině operačního systému C, C++, assembler, či jejich kombinací, nebo využití dávkových souborů. Na vyšších vrstvách se pak často tvoří malware pomocí skriptovacích jazyků, jazyka python, C# a dalších.

9.8.1.1 Procesor a paměť

V následujících kapitolách budou probírány metody exploitace. Jelikož většina technik souvisí se změnou toku programu, která je způsobena narušením paměti, je potřeba se na paměť blíže zaměřit.
Aby mohl program vykonávat svoji činnost, potřebuje paměť k odkládání svých proměnných. Pamětí se nemyslí pouze operační paměť počítače, ale i procesor má svá vlastní paměťová místa, která se nazývají registry. Některá slouží jako pracovní registry, další odkazují na místa v paměti - především na zásobník, jiná uchovávají příznaky.
Každý proces získá přístup do paměti, která je rozdělena na segmenty[3][10]:

- Text - Slouží pro uchování kódu programu, velikost je pevná.
- Data - Segment ukládá globální proměnné programu, velikost je pevná.
- BSS - Uchovává neinicializované proměnné, velikost je pevná.
- Halda (anglicky "heap") - Část určená k dynamické alokaci proměnných. Jelikož kompilátor předem nezná velikost požadovaných proměnných, používá se k odkazování do této části ukazatelů. Stanou-li se dynamicky alokované proměnné nepotřebné, musí být uvolněny, aby nedocházelo k úniku paměti. Velikost haldy je tedy proměnlivá a roste od nejnižší adresy k nejvyšší.
- Zásobník (anglicky "stack") - Velikost je opět proměnlivá a plní se naproti haldě - od nejvyšší adresy po nejnižší. Na zásobník se ukládají lokální proměnné pro každou

volanou funkci, ale především i návratová adresa. Každému bloku náležícímu k určité funkci se říká zásobníkový rámec.
- Prostředí - Paměťový segment ukládající parametry předané z běhového prostředí a kopie systémových proměnných.

9.8.2 Akce

K čemu by byl virus, který by nic nedělal? Útočník při programování vkládá do malware kódy, které budou na cílovém systému plnit jeho přání. Jak již bylo řečeno, možnosti mohou být takové, jaké dovoluje běhový systém.
Zde jsou uvedeny některé akce, které útočník po malware požaduje:

- Vytváření lokálních kopií
- Šíření svých kopií
- Aktivace po startu systému
- Ukrytí se v systému
- Vytvoření zadních vrátek
- Zahájení síťové komunikace
- Stažení dalšího malware
- Špionáž
- Editaci či mazání dat
- Provést útok
- Rozeslání spamu
- Mnoho jiného

Obsahově by bylo velmi rozsáhlé popisovat různá prostředí a v nich způsob provedení jednotlivých akcí. Práce proto bude zaměřena na systémy Microsoft Windows XP, Windows 7 a Windows 8. OS nabízí nepřeberné množství funkcí, které programátorovi zpřístupňuje pomocí API rozhraní - pod systémy Microsoft Windows se toto rozhraní označuje jako Windows API, Win32 API či WinAPI. Pomocí Windows API tedy může programátor realizovat činnosti jako práce se soubory, práce se sítí (WinSocket), sběr informací, ovládání hardware, přenastavení systémových registrů a mnohé další. Mezi důležitou součást patří také schopnost spustit jiné procesy.
Je-li pod operačním systémem spouštěn nový proces, obdrží práva uživatele, který jej vyvolal. Toto může být pro útočníka velmi limitující. Při volání důležitých funkcí jsou systémem kontrolována oprávnění a podle toho je akce povolena či zakázána. Dnešní souborové systémy také ukládají u souborů a složek značky, podle kterých systém hodnotí, zda k nim má proces přístup. Chce-li útočník zajistit, aby jeho malware mohl využít co nejvíce funkcí, bude potřeba získat co možná nejvyšší oprávnění.
Získání oprávnění:

- Spuštění privilegovaným uživatelem - má-li útočník štěstí, je jeho malware spuštěn pod uživatelským účtem s plným oprávněním. Program může být napsán způsobem, který jej registruje v systému a je spouštěn při každém startu. Stačí tedy, aby malware získal administrátorská práva pouze jednou a pak se registroval jako spouštěný s těmito právy.
- Znalost hesla - získá-li útočník heslo k privilegovanému účtu, může hesla využít ke zvýšení oprávnění.
- Využití chyb - je jasné, že při běžném provozu je spouštěno mnoho procesů s různým oprávněním. Nalezne-li útočník využitelnou chybu v procesu s vyšším oprávněním, může ji využít a získat práva napadené aplikace. Tomuto postupu se také říká "lokální exploitace".

9.8.3 Lokální exploit

Slovem exploit je označován specifický program (nebo sekvence dat), který využije chybu systému, či jiného programu, ke svému prospěchu. Cílem technik je převzetí kontroly nad tokem programu.[1]
Exploity mohou být děleny dle komunikace se zranitelným software na lokální a vzdálené. Lokální exploit se používá k získání privilegií aplikace, na kterou je exploit použit. Vzdálené exploity budou probírány později.
Většina exploitů souvisí s narušením paměti - v programu má každá alokovaná proměnná v paměti přidělené své místo a velikost. Pokusí-li se však proces uložit větší proměnnou, než pro kterou je paměťové místo vyhrazeno, přepíší se i data následující. Tato situace není kompilátorem ani systémem kontrolována, protože by to znamenalo výrazné zpomalení systému - integrita proměnné by musela být kontrolována při každém jejím ukládání do paměti.

9.8.3.1 Přetečení zásobníku

Každému spouštěnému procesu je přidělena část paměti obsahující segment typu zásobník. Jak již bylo zmíněno, zavoláním funkce z programu se vytvoří na zásobníku nový zásobníkový rámec, který obsahuje také návratovou adresu z funkce. Cílem útoku je provést přetečení do této návratové adresy a nastavit hodnoty na útočníkem definované paměťové místo. Pachatel také může do paměti vložit vlastní instrukce a přinutit program udělat zamýšlenou akci. Těmto vloženým instrukcím se říká shellkód.

9.8.3.2 Přetečení na haldě

Paměťový segment halda neslouží k řízení toku programu, ale lze jej taktéž využít k útoku. Na haldu se ukládají dynamicky alokované proměnné, které mohou nést důležitá data. Například cestu k souboru či síťovou adresu. Podaří-li se nalézt místo v programu, které je zranitelné na přetečení, může útočník na následující paměťové pozice zapsat data, která požaduje. Tímto způsobem kompromituje například adresu souboru, do kterého se budou ukládat důležitá data nebo síťovou adresu na kterou program bude vytvářet připojení.

9.8.3.3 Přetečení v segmentu bss

Do tohoto paměťového segmentu se ukládají globální proměnné. Globální proměnnou může být i ukazatel na funkci. Podaří-li se útočníkovi kompromitovat hodnotu proměnné, může ji pozměnit na jinou, požadovanou funkci. Ukazatel tak odkáže vykonávání programu do jiné funkce.
Existují i různé další taktiky, jak procesu podvrhnout data - programátor si však musí nastudovat všechny vstupní cesty a podle toho nejlépe zvolit využitelnou metodu. Hledání nových chyb je často běh na dlouho trať, jelikož ne vždy musí program konkrétní chyby obsahovat a ne vždy lze nalezené chyby využít. Hacker tak tráví mnoho času nad ladícími programy a kódem programu, který chce napadnout.
Ne vždy potřebuje útočník nalézt v programu úplně novou chybu, ale vystačí si již s chybou, kterou objevil někdo jiný. Existují veřejné databáze zneužitelných chyb, které útočníkovi ušetří mnoho času. Tyto chyby bývají většinou zveřejněny legálním způsobem a to až po určitém čase, kdy autor postiženého programu již vydal opravenou verzi svého produktu a uběhla dostatečně dlouhá doba potřebná k aktualizaci na klientských stanicích.
O těchto databázích, a jejich použití, bude řeč v kapitole věnované penetračnímu testování.

9.8.4 Ukrytí

Je-li útočníkovým záměrem svůj kód v cílovém prostředí ukrýt, bude vyžadováno, aby byl co nejméně nápadný pro uživatele a bezpečnostní programy. Opět platí, že možností ukrytí je mnoho a záleží na konkrétním typu malware a kreativitě útočníka.

9.8.4.1 Ukrytí před uživatelem

Je patrné, že míra odhalení také často závisí na znalostech uživatele. Kód se v systému bude projevovat svými akcemi a také bude někde přechováván.

- Aktivace kódu po spuštění systému - Velká část malware se registruje do systému tak, aby byl jeho kód spuštěn po načtení systému či přihlášení uživatele. OS Microsoft nabízí několik cest, jak toho dosáhnout. Vložení kódu či odkazu do složky "po spuštění", vložení záznamu do registru, aktivace pomocí plánovače, nebo registrace kódu jako systémové služby. Útočník může využít i sofistikovanějších metod - připojit malware na některý ze spouštěných procesů, tak bude spuštěn současně s tímto procesem. Nemusí se vždy jednat o systémovou službu, připojit se může i k programu, který bude uživatelem patrně často spouštěn ručně.
- Uložení - většina malware je naprogramována způsobem, kdy uloží svůj kód na datový nosič systému. Pachatel většinou nevolí místo uložení tak, aby se kód nacházel na zřetelném místě. Často je ukryt v adresářové struktuře mimo uživatelem běžně používané složky. I zde existují pokročilejší metody, jak kód skrýt. Může se jednat o připojení kódu malware na jiný spustitelný soubor či umístění do specifické oblasti na disku - příkladem může být Alternate Data Streams, který je podporován souborovým systémem NTFS. Implementován byl z důvodů kompatibility se souborovým systémem HFS, současně však vytvořil vhodné místo i pro malware. Běžně se soubory ukládají do primárního datového proudu, útočník ovšem použije přesměrování toku a nový soubor pojmenuje ve formátu "původní:nový". Tím zůstane původní soubor na svém místě v primárním proudu, ale v ADS se vytvoří soubor "nový". Prochází-li uživatel své soubory, vidí právě ty, které jsou v proudu primárním, soubory v ADS ale již nevidí. Dokonce se nezobrazí ani jiná velikost původního souboru.
- Správce procesů - Správce procesů OS Windows zobrazuje aktivní procesy všech uživatelů. Existují možnosti, jak nově vytvořený proces skrýt. Mezi nejvíce používané patří registrovat proces jako službu, či připojit program na jiný. Dokonce i v případě, že je aplikace spuštěna z ADS, je ve správci viditelná jako proces z primárního proudu. Pachatel může uživatele také oklamat tím, že dá programu jméno vyhlížející jako systémový proces nebo důvěryhodná aplikace.
- Akce - uživatel může rozpoznat malware podle jeho projevů. Situace, kdy malware smaže uživateli data, jej rychle prozradí. Existují však i jiné, méně nápadné, projevy - zvýšená aktivita procesoru, delší odezva při spuštění aplikace, neočekávané pády, zvýšená a nepovolená síťová komunikace, otevřený port pro příchozí spojení.

9.8.4.2 Ukrytí před bezpečnostním software

Právě antivirové programy jsou ty, které chrání uživatele před usídlením a spuštěním nákazy. Cílem útočníka je tento software vyřadit z provozu, nebo před ním svůj malware ukrýt - mezi nejvíce užívané techniky patří matení kódů a kódování.
Podle maskovací techniky se viry dělí do skupin:

- Kódované viry - Tvůrce viru využije algoritmu, kterým zakóduje tělo viru a do jeho vstupní funkce vloží dekódovací algoritmus ("dekryptor"). Dekryptor musí být stále

stejný, aby mohl tělo viru dekódovat. Často je využito i klíče, který každou další kopii viru dělá jedinečnou[6].

- Oligomorfní viry - Vylepšuje kódované viry tím, že ve svých kopiích mění i svůj dekryptor a každá následující generace je odlišná nejen svým tělem, ale i dekryptorem.
- Polymorfní viry - Pokročilejší oligomorfní viry - přidávají nadbytečné instrukce, různé formy kódování, vícevrstvé kódování, v neposlední řadě také prohazují pořadí instrukcí, ale způsobem, aby výsledná akce byla ekvivalentní. Ve výsledku mohou vznikat až milióny různých exemplářů.
- Metamorfní viry - Neobsahují dekryptor, ani nemají konstantní tělo a jsou schopné vytvářet odlišné kopie[4]. Nevytváří žádná konstantní data či struktury, pomoci kterých by byly odhaleny. Metamorfní viry mohou být například distribuovány ve formě překladače a zdrojového kódu, který je zakódovaný. Při kompilaci se současně dekóduje i zdrojový kód, jsou přidávány instrukce, mění se jejich pořadí a zdrojový kód je znova kódován, ale jiným klíčem.
- Retro viry - Jejich úkolem je zaútočit na bezpečnostní software a pokusit se jej vyřadit či modifikovat tak, aby neblokoval následné akce. Tím připraví cestu pro další malware.

9.8.5 Šíření

Autor malware často požaduje, aby bylo zajištění jeho šíření. Malware je většinou naprogramován tak, aby po spuštění vytvořil svou kopii na lokálním systému, která je znova spouštěna při každém zapnutí stroje. Často se malware snaží šířit i na další vzdálené systémy skrze přenosná média či síť.

9.8.5.1 Šíření na lokálním systému

- Kopírovací metoda - malware má definováno, že bude vytvářet své nové kopie na disku počítače a také na výměnná média při jejich připojení.
- Přepisovací metoda - malware prochází adresáře a hledá soubory zvoleného typu, ty přepíše svým tělem. Původní soubor je tedy trvale poškozen.
- Parazitická metoda [12] - Na rozdíl od předešlé metody soubor nepoškodí, ale připojí k němu svůj kód. Malware se může připojit před kód původního programu, za program, ale také vložit mezi části původního programu. Vir pozmění informace v PE hlavičce, aby zajistil svou aktivaci.

9.8.5.2 Šíření na vzdálené systémy

Vyměnitelná média - Stejný princip jako v předešlé části. Malware zkopíruje kód na vyměnitelné médium.

- Síťová komunikace - Mnoho služeb umožňuje síťovou komunikaci. Pro útočníka je tato metoda výhodná, jelikož k rozesílání může zpravidla využít účet a kontakty oběti. Příjemce pak obdrží zprávu, jehož adresátem je napadený uživatel. Malware pak často mění název souboru.
- Elektronická pošta - Často používaný nástroj. Malware prohledá adresáře poštovních programů a na nalezené adresy se rozešle - nejčastěji jako příloha, či formou odkazu.
- Služba instantních zpráv - Různé služby nabízí možnost zasílání zpráv v reálném čase. I tyto služby jsou napadnutelné. Malware se rozešle na kontakty ve formě souboru či odkazu na webové stránky.
- Sdílení adresářů - Malware může přenést svůj kód do vzdálených adresářů, či sdílet sám sebe.
- Jiné - Malware může využít i jiné služby ke svému přenesení. V neposlední řadě může nést i vzdálený exploit, který zneužije bezpečnostní mezery.

Možností šíření se nabízí celá řada, ne všechny cesty jsou proto jmenovány.

9.8.6 Komunikace

V tomto bodě má útočník provozuschopný malware, který po napadení cílového systému vykoná předdefinované akce. Ale to je vše. Chce-li mít autor možnost ovládat svůj program i po té, co je již aktivní na cílovém stroji, musí do něj implementovat mechanismy, které to povolí. Je jasné, že cílový stroj pak musí být dosažitelný skrz internet (či vnitřní síť).
Komunikace v internetu je založena na modelu TCP/IP, proto je potřeba implementovat technologii, která tuto komunikaci umožní. Na úrovni OS Windows ji implementuje rozhraní Winsock. Socket definuje koncový bod komunikace - aplikace vytváří sockety k zahájení odchozího spojení a také k naslouchání na spojení příchozí[7].
Komunikace se dělí dle toho, kdo spojení navazuje:

- Útočník - malware je implementován způsobem, kdy naslouchá na zvoleném portu. Útočník se vzdáleně na předvolený port připojí a může začít komunikace. Komunikace je obousměrná,takže zdroj i cíl mohou současně na sokety zapisovat i číst. Je patrné, že přenos sice funguje a malware bude zprávy číst, ale to je vše. Proto musí být implementován komunikační protokol, kterému budou rozumět obě strany. Pak již bude umět malware rozpoznat požadavek na akci, splnit jej a zaslat útočníkovi odpověď. Toto řešení má dvě velké nevýhody - je-li cílová stanice za firewallem, bude sice naslouchat, ale požadavek na spojení bude firewallem zakázán, dále je nutné, aby měla stanice stále stejnou IP adresu, či platný záznam v DNS službě. Proto je toto řešení uplatnitelné pouze u serverů, které mají nastavené směrování na pevnou veřejnou IP adresu.
- Oběť - v tomto případě útočník zpřístupní server, který naslouchá na své adrese a definovaném portu. Do těla malware je pak tato adresa (či doménové jméno) vložena. Malware může být nastaven způsobem, že se připojí k serveru a udržuje spojení aktivní, a nebo je spojení navazováno jen v určitých časových intervalech a malware si stáhne novou definici příkazů či svou aktualizaci. Útočník může chránit svou identitu tak, že při každé aktualizaci se změní také adresa na které proběhne následná aktualizace. To ale přináší práci s neustálým přesouváním serveru. Jednodušší způsob je využít k přístupu na server anonymizační proxy ze strany pachatele - proxy serverem může být i jiný napadený stroj.

9.9 Penetrace

Cíl je definován - přenést malware na stroj oběti a aktivovat. Přichází čas, kdy budou ověřeny schopnosti a kreativita útočníka. Penetrace, nebo-li vniknutí cizího kódu, je vrcholnou, a většinou také nejtěžší, částí celého procesu. Je běžné strávit většinu času při tvorbě malware právě nad studiem cílového systému a hledáním efektivních cest k přenesení a spuštění hrozby.
Existuje sice množství cest, ale každá je jinak nápadná pro uživatele, časově náročná, přístupná, ale také ne každá z nich zaručí výsledek. Možností také je, že sebelepší malware skončí právě u tohoto kroku. Budou popsány různé cesty, které lze při šíření malware využít:

9.9.1 Fyzicky přístupný stroj

Pro útočníka je tato situace nejvíce přívětivá, ale také často nedostupná. Obzvláště, je-li útočníkovým cílem širší skupina strojů. Útok na lokální stroj není příliš složitý - existuje více metod, ale zpravidla se využívá odděleného systému na datovém nosiči útočníka. Ten se zavede namísto původního systému a následně může provádět prakticky jakékoli úpravy na discích

lokálního počítače, včetně zkopírování souborů obsahující hesla (v hash podobě). Obrana proti takovému útoku nespočívá ve vytvoření hesla pro BIOS a nastavení priorit zavádění - útočník může lehce nastavení BIOS obnovit. Obrana spočívá v zašifrování disků a také především v zamezení onoho fyzického přístupu neoprávněným osobám. Je možné, že útočník také zkopíruje celý obraz zašifrovaného disku - bude-li ovšem užito bezpečného šifrovacího algoritmu a silného hesla, neměl by být útočník schopen heslo dekódovat v rozumném čase.

9.9.2 Vzdálený exploit

Chce-li být útočník co možná nejméně nápadný, zvolí jako další postup testování konkrétních služeb, jsou-li náchylné k exploitaci a následnému rozšíření viru do systému.
V minulých kapitolách bylo napsáno, že existují různé databáze uchovávající exploity k již nalezeným zranitelnostem[31][32] a to jak lokálním, tak i vzdáleným. Chce-li pachatel vzdálený exploit provést, ale současně i strávit co možná nejméně času nad hledáním zranitelností, je to pravé místo, kde začít. K dispozici je také opensource produkt "Metasploit", jehož cílem je usnadnit vyhledávání zranitelných míst a použití exploitů.[4] Metasploit obsahuje svou vlastní databázi exploitů a nálož (anglicky "payload"). Nálož je kód, který bude po úspěšné penetraci spuštěn.
Útočník si vytvoří konfiguraci počítače oběti ve svém vlastním prostředí a nainstaluje požadované služby tak, aby se verze shodovaly s cílovým strojem. Nyní může začít testovat zranitelnost - ve vnitřní databázi produktu metasploit vyhledá exploity k testované službě a vyzkouší, zda je na ně služba náchylná. V textu již bylo zmíněno, že zranitelnosti se do databází dostávají až po delším čase, proto je možné, že služba (v testované verzi) nebude na žádnou z těchto zranitelností náchylná. Pachatel pak může využít ostatních databází - je možné, že exploit z databáze bude potřeba upravit do podoby, aby jej Metasploit dokázal interpretovat. I zde může nastat situace, že exploit nemusí být aplikovatelný.
Útočník se tedy rozhodne vytvořit exploit vlastní. Cenným zdrojem informací pro něj mohou být přímo webové stránky produktu, na který se rozhodne zaútočit. Autoři programů často uvádí seznam oprav, které provedli v nové verzi a to včetně těch bezpečnostních. Je-li tedy vzdálená služba neaktuální, může se útočník pokusit vytvořit exploit na tuto zranitelnost a napadnout tak cílový stroj.
Pachatel se může pokusit najít zranitelnost vlastními silami. Je potřeba si uvědomit, že oproti lokální službě je nyní komunikace omezena pouze na spojení přes sockety. I tak může systém obsahovat chyby, které často souvisí s formátováním řetězce či přetečením paměti. Mezi nejznámější chyby způsobené neošetřením vstupu patří Cross-site scripting a SQL injection - ty sice neumožní spuštění binárního kódu, ale mohou útočníkovi pomoci při získání cenných informací. Obzvláště, dostane-li se pomocí SQL injection k tabulce s hesly.
Je-li služba náchylná na útok přetečení paměti, může tvůrce exploitu využít techniky popsané v kapitole "lokální exploit" ovšem s rozdílem, že využije shellkód tak, aby zprostředkoval stažení malware.

9.9.3 Získání přihlašovacích údajů

V případě, že na cílovém stroji běží služba, která umožňuje vzdálené přihlášení, může se k ní útočník pokusit získat přihlašovací údaje. Většinou je vzdálená správa povolena na serverech, na osobních počítačích méně. Pachatel se ovšem může pokusit změnit (pomocí vzdálené správy) konfiguraci na směrovačích a přepsat adresy DNS serverů. Odešle-li oběť dotaz na útočníkův DNS server, bude mu zaslána jiná IP adresa. Při prohlížení webových stránek to bude mít za následek načtení jiné stránky. Ta může obsahovat škodlivý kód zneužívající chyby prohlížeče nebo nabídnout malware ke stažení.

K získání přihlašovacích údajů může vést mnoho cest. Útočník se může pokusit o uhodnutí hesel slovníkovým útokem, či útokem hrubé síly. Vychází-li útočník z předpokladu, že uživatel používá stejné přihlašovací údaje i na jiných službách, může se pokusit také o jejich získání.

9.9.4 Metody sociálního inženýrství

Sociálnímu inženýrství byla věnována celá kapitola, proto zde budou rozebírány pouze postupy související s penetrací malware. Do této metody spadají techniky, které přimějí uživatele, aby malware sám spustil a nebo dopomohl k jeho spuštění. Využívá se především maskování - malware se může vložit do jiného programu, či se připojit na spustitelný soubor, jak bylo popisováno výše. Často autoři pouze mění název souboru a před koncovku vloží další příponu jiného formátu. Útočník pak takovýto soubor odešle elektronickou poštou, předá na výměnném médiu, či vystaví na stránky, kam oběť přiláká.

9.10 Životní cyklus

I malware má svůj životní cyklus. Může být rozdělen na jednotlivé fáze: vypuštění, rozšíření, aktivita, odhalení, úpadek, zánik. Dělení ovšem není vždy aplikovatelné - příklad může být malware, který se nepodaří rozšířit, malware který provede svůj update a znova musí dojít k jeho odhalení, či malware který je dosud neobjeven a stagnuje.
Pro útočníka není vždy lehké udržet svůj malware v oběhu. Je-li detekována nová hrozba, kterou antivirové programy dosud nerozeznávají, výrobci přidají její definici do databáze svých antivirových programů. Uživatelé si nové definice stáhnou a antivir již bude moci malware odhalit. Podobná situace může nastat i u malware, jež využívá určité chyby v systému. Autor implementuje záplatu a vydá novou verzi programu. Po stažení nové verze již není možné tuto mezeru využít a malware přejde do fáze úpadku či stagnace. Chce-li být pachatel úspěšný, musí hledat nové způsoby, jak vylepšit svůj malware, či provádět své akce velmi nepozorovaně. Je-li totiž na stroji oběti chyba opravena a malware eliminován, ztrácí útočník nad takovýmto strojem kontrolu.

10 Příloha A - důležité příkazy, klávesové zkratky a konzole správy operačního systému Windows

Pro každého bezpečnostního analytika ať už je na správné nebo špatné strany barikády je nezbytné dokonale znát systémy které chrání (nebo napadá). Ne úplně všechny následující příkazy a zkratky se přímo týkají bezpečnosti systému Windows ale jak si už čtenáři možná stihli všimnout důvody úspěchu hackerů leží právě v tom že dokáží zkombinovat a využít ve svůj prospěch všechny možné části systému včetně těch u kterých není na první pohled zjevná souvislost s bezpečností.

Klávesové zkratky operačního systému
Windows Logo + D: zobrazení plochy
Windows Logo + M: Minimalizace všech oken
Windows Logo + shift + M: maximalizace oken
Windows Logo + R: dialog Spustit
Windows Logo + E: Tento počítač
Windows Logo + Break: Vlastnosti systému
Windows Logo + L: uzamknutí klávesnice
Windows Logo + F1: nápověda systému Windows

Proměnné Windows:
Lze využít v jiných příkazech jako zástupce místo skutečné cesty, např. „cd %systemroot%“.
%allusersprofile%: domovský adresář všech uživatelů
%homedrive%: domovský disk (např. c:\)
%userprofile%: domovský adresář aktuálně přihlášeného uživatele
%temp%: složka dočasných souborů
%systemroot%: složka se systémovými soubory

Zkratky pro dialogové boxy s nastavením systému:
access.cpl: Možnosti usnadnění
hdwwiz.cpl: Průvodce přidáním nového hardware
appwiz.cpl: Přidat nebo odebrat programy
timedate.cpl: Datum a čas – vlastnosti
desk.cpl: Vlastnosti desktopu
inetcpl.cpl: Vlastnosti Internetového připojení
main.cpl: Vlastnosti myši
ncpa.cpl: Připojení k místní síti
powercfg.cpl: Možnosti napájení
intl.cpl: Místní a jazykové nastavení
mmsys.cpl: Zvuky a zvuková zařízení
sysmd.cpl: Vlastnosti systému
nusrmgr.cpl: Uživatelské účty
firewall.cpl: Firewall
wscui.cpl: Možnosti zabezpečení

Příkazy příkazového řádku:
Arp: program pro správu Address Resolution Protocol
At: naplánované spuštění libovolného příkazu nebo souboru
Bootcfg: zobrazení a nastavení spouštění více operačních systémů („bootcfg /default /1“ nastaví jako výchozí volbu instalaci operačního systému s ID 1)
Calc: spuštění kalkulačky

Charmap: mapa znaků
Chkdsk: kontrola disku
Cipher: šifrování souborů a adresářů
Cleanmgr: vyčištění pevného disku (dočasné soubory, komprese starých souborů, koš,..)
Clipbrd: prohlížeč schránky
Cls: vyčištění obrazovky
Cmd: otevře nový příkazový řádek
Comp: porovnání dvou souborů nebo adresářů
Control: otevře Ovládací panely
Date: zobrazí/nastaví datum
Dcomcnfg: konzola Služba komponent (konfigurace Služby komponent, Prohlížeč událostí, Služby)
Dxdiag: Nástroj pro diagnostiku rozhraní DirectX
Explorer: průzkumník (lze využít například tehdy když dojde k pádu uživatelského rozhraní systému - například když omylem „odpálíme“ proces explorer.exe – ale když je ještě stále možné pustit „Správce úloh systému Windows“ pomocí kláves CTRL+SHIFT+ESC. Pak stačí vybrat volbu Soubor/Spustit a v něm zadat „Explorer.exe“).
Fc: porovnání dvou souborů
FTP: klient protokolu FTP v příkazovém řádku (ukončení příkazem „quit")
Hostname: zobrazí jméno počítače
Ipconfig: zobrazí konfiguraci připojení k síti
Mem: zobrazí využití paměti
Net: správa síťových prostředků (účty, počítače, soubory, skupiny, služby, sdílené prostředky). Tento příkaz se skládá ze dvou částí, slova net + druhého slova které může být: Accounts, Computer, Config, Continue, File, Group, Help, Helpmsg, Localgroup, Name, Pause, Print, Send, Session, Share, Start, Statistics, Stop, Time, Use, User, View. Nápovědu k jednotlivým podpříkazům lze získat např. „net help accounts“. Jedná se o velmi důležitý příkaz pomocí kterého lze provádět komplexní správu počítače – například z příkazové řádky přidávat uživatele (net user jmeno heslo /add).
Netstat: zobrazí síťovou statistiku („netstat –a“ zobrazí všechna spojení, „netstat –b“ zobrazí název binárního souboru který spojení otevřel, „netstat –n“ zobrazí čísla portů místo symbolických názvů).
Nslookup: nástroj příkazového řádku pro dotazování se jmen serverů na DNS serveru (po spuštění zadávat jména domén + Enter)
Path: systémová proměnná zobrazující cesty k často používaným souborům
Perfmon: sledování výkonu
Ping: zjištění odezvy vzdáleného serveru
Regedit: editor registrů
Regsvr32: registrování dll souborů do systému
Route: správa síťových směrovacích tabulek
Runas: spustí program jako jiný uživatel
Rundll32: nahraje DLL soubor
Set: Nastavení místní proměnné
Sfc: Kontrola systémových souborů („sfc /scannow“ provede okamžité skenování systémových souborů zda nejsou změněny)
Shutdown: vypne počítač
Sndrec32: záznam zvuku
Sndvol32: nastavení hlasitosti
Sysedit: editor systémových souborů (win.ini,system.ini, config.sys, autoexec.bat)
Systeminfo: informace o systému (verze operačního systému, service packy, záplaty, doba provozu systému, IP adresy). Důležitá je pro nás především informace o nainstalovaných záplatách. V kombinaci s přesměrováním do příkazu findstr pomocí roury je možné si odfiltrovat pouze konkrétní číslo záplaty a tím rychle zjistit, zda je v systému přítomná.

Taskmgr: správce úloh
Telnet: textový klient pro připojení k vzdálenému terminálu
Tracert: Výpis trasy k zadanému server
Ver: verze operačního systému
Winmsd: systémové informace

Konzole MMC:
Tyto konzole lze spustit z příkazového řádku zadáním jejich jména, např. „services.msc“. Většina z nich je dostupná ze všech adresářů protože jsou v proměnné PATH, některé ale nemusí být (např. iis.msc). Ty je nutné najít a spustit z adresáře ve kterém leží.
certmgr.msc: správa obsahu úložišť certifikátů pro uživatele, službu a počítač.
ciadv.msc: správa služby indexing service (indexovaní služba, stará se o indexování souborů pro jejich vyhledávání)
comexp.msc: správa komponent (je v %systemroot%/system32/com/)
compmgmt.msc: konzole správy počítače (prohlížeč událostí, sdílené složky, uživatelé a skupiny, správce zařízení, úložiště, služby a aplikace)
devmgmt.msc: správce zařízení
dfrg.msc: defragmentace disku
diskmgmt.msc: správa disků
fsmgmt.msc: sdílené složky
eventvwr.msc: prohlížeč protokolu událostí systému a jednotlivých aplikací
gpedit.msc: zásady skupiny
iis.msc: správa webového serveru Internet Information Services, dostupné také přes Start/Nastavení/Ovládací panely/Nástroje pro správu/Internetová Informační Služba
lusrmgr.msc: místní uživatelé a skupiny
ntmsmgr.msc: vyměnitelné úložiště
perfmon.msc: sledování výkonu (paměť, disk, procesor)
secpol.msc: místní nastavení zabezpečení
services.msc: služby systému Windows
wmimgmt.msc: správa služby WMI (Windows Management Instrumentation)

11 Příloha B – Odkazy

http://www.soom.cz
Jedna z nejznámějších českých stránek s bezpečnostní tématikou. Jsou zde články, recenze literatury, nástroje ke stažení, rozsáhlé diskusní fóra v češtině a také sekce BugTraq ve které uživatelé serveru zveřejňují jimi nalezené chyby na konkrétních serverech (nejčastěji ve webových aplikacích).
https://kinqpinz.info/iso/
Web s bohatými a jinde těžko dostupnými zdroji informací a nástrojů. V sekci iso jsou ke stažení velmi zajímavé distribuce volně šiřitelných operačních systémů:
Anonym-OS (bootovací LiveCD na bázi OpenBSD konstruované na maximální možnou anonymitu uživatele),
Auditor Security Collection (Live distribuce založená na Knoppixu, obsahující více než 300 bezpečnostních nástrojů),
BackTrack: LiveCD pro bezpečnostní auditoring. Velmi populární distribuce.
Damn Small Linux (50 megabytový Live image bootovatelný z CD, HDD nebo USB zařízení. Cílem této distribuce je natěsnat co nejvíce nástrojů do 50 mB limitu. Obsahuje téměř kompletní výbavu desktopu plus SSH/FTP/HTTPD servery.),
Damn Vulnerable Linux (živá distribuce určená pouze pro vzdělávací účely – byla vytvořena tak aby se jednalo o co nejzranitelnější systém. Kromě schválně přidaných chyb obsahuje také bezpečnostní nástroje pomocí kterých se tyto chyby dají najít a zneužít a výukové materiály včetně návodů na obsažené chyby poskytnutých členy komunity crackmes.de).
Hacking – The Art of Exploitation Live CD: doprovodné cd ke knížce Hacking – The Art of Exploitation, 2nd Edition, autor Jon Ericsson (vyšla také v češtině).
Hakin9 Live: cd s tutoriály k magazínu Hakin9 (dostupný také v české mutaci i když s příšerným překladem.
nUbuntu: bezpečnostní distribuce založená na populárním Ubuntu Linuxu.
Russix: Distribuce založená na Slaxu, kompletně věnovaná auditování Wi-Fi sítí.
http://insecure.org
Domovská stránka „Network Mapper“ – nmap scanneru, jednoho z nejlepších nástrojů pro skenování portů.
http://www.immunitysec.com/products-canvas.shtml
Domovská stránka Immunity CANVAS, frameworku obsahujícího stovky exploitů a prostředí pro jejich testování a vývoj. Jedná se o hackerský nástroj v hodnotě tisíců dolarů.
http://www.gleg.net/products.shtml
Domovská stránka VulnDisco Pack Professional (dříve známý jako Argeniss Ultimate 0day Exploits). Stovky exploitů použitelných jako moduly do CANVAS frameworku. Jedná se o 0-day exploity nedostupné pro veřejnost.
http://www.sensepost.com/
Domovská stránka společnosti SensePost, nezávislé společnosti specializované na informační bezpečnost. Společnost je známá především díky svým činnostem souvisejícím s využíváním vyhledávače Google z hlediska bezpečnosti. Mezi jejich nejznámější nástroje patří Wikto, DiBiBLAH a E-Or nahrazený nyní Suru.
http://www.crackmes.de/
Největší světová „crackme“ a „reverseme“ komunita. Nabízí přes 2000 crackmes cvičení včetně řešení, má více než 13 000 členů, bohaté fóra a videoportál.
http://www.redoracle.com
Hackerská skupina s vlastní bezpečnostní distribucí, aktuálním archivem nástrojů a denně updatovanou sekcí s veřejnými exploity a vulnerabilities.
http://www.ethicalhacker.net/
Články s recenzemi bezpečnostních nástrojů (lze je použít i jako návod pro složitější nástroje ke kterým není dostatečná dokumentace) a obecně témata etického hackingu.

http://darkc0de.com/cgi-bin/proxies.py
Real-timově updatovaný seznam proxy serverů.
http://www.windowhaxor.net/
„Microsoft Windows Hacking Portal“ – blog počítačového nadšence zveřejňující užitečné nástroje a tipy jak upravovat, spravovat, zabezpečovat a prolamovat zabezpečení OS Windows.
http://securitylabs.websense.com/content/blogs.aspx
Blog o bezpečnostních problémech hostovaný na stránkách společnosti Websense.
http://googlesystem.blogspot.com/
Blog o neoficiálních novinkách a funkcích impéria Google.
http://www.doxpara.com/
Osobní stránka špičkového hackera Dana Kaminskyho, autora Paketto Keiretsu (extrémně rychlý scanner) a výzkumníka v oblasti bezpečnostních aspektů TCP/IP a Internetu. K dispozici jsou jeho přednášky z Black Hat konferencí – jedná se o hacky a kombinace technologií které jsou na špičce současného poznání.
http://airdump.cz/
Rozsáhlý český bezpečnostní portál – novinky, tutoriály, blogy, wiki fórum.
http://www.securitycompass.com/exploitme.shtml
Stránky projektu ExploitMe, také zdroj White paper materiálů například o buffer overflow.
http://rgaucher.info/planet/Security_Thoughts/
Blog s příklady zranitelností apache
http://httpd.apache.org/security/vulnerabilities_13.html
http://httpd.apache.org/security/vulnerabilities_20.html
http://httpd.apache.org/security/vulnerabilities_22.html
Oficiální stránky http serveru Apache – dokumentace k jednotlivým verzím se seznamem nalezených a záplatovaných chyb. Velmi užitečná stránka pokud naleznete server na kterém běží některá z neaktuálních verzí serveru.
http://ghh.sourceforge.net/
Google Hack Honeypot je reakcí na nový druh zlovolného síťového provozu – hackerského průzkumu pomocí vyhledávačů.

http://www.doxpara.com/
Osobní blog Dana Kaminskeho, objevitele kritické chyby v DNS systému v roce 2008.

12 Doporučená literatura

1. ERICKSON, Jon. Hacking: umění exploitace. 2., upr. a dopl. vyd. Překlad Jan Pokorný. Brno: Zoner Press, 2009, 544 s. ISBN 978-80-7413-022-9.
2. SELECKÝ, Matúš. Penetrační testy a exploitace. 1. vyd. Brno: Computer Press, 2012, 303 s. ISBN 978-80-251-3752-9.
3. HARRIS, Shon, Allen HARPER, Chris EAGLE, Jonathan NESS a Michael LESTER. Hacking: manuál hackera. 1. vyd. Praha: Grada, 2008, 399 s. ISBN 978-80-247-1346-5.
4. SZOR, Peter. Počítačové viry: analýza útoku a obrana. Vyd. 1. Brno: Zoner Press, 2006, 608 s. ISBN 80-868-1504-8.
5. GRAVES, Kimberly. CEH: certified ethical hacker study guide. Indianapolis, Ind.: Wiley Pub., 2010, xxxvii, 392 p. ISBN 978-0-470-52520-3.
6. HEINIGE, Karel. Viry a počítače. Praha: Knihy iDNES, 2001. ISBN 80-86097-74-9.
7. Malware analyst's cookbook and DVD: tools and techniques for fighting malicious code. Indianapolis: Wiley, c2011, xxvi, 716 s. ISBN 978-0-470-61303-0.
8. JAMES, Lance. Phishing bez záhad. 1. vyd. Praha: Grada, 2007, 281 s. ISBN 978-80-247-1766-1.
9. CLARKE, Nathan. Proceedings of the Sixth International Symposium on Human Aspects of Information Security: Crete, Greece, 6-8 June 2012. United Kingdom: University do Plymouth, 2007, iv, 216 pages. ISBN 978-184-1023-175.
10. MAREK, Rudolf. Učíme se programovat v jazyce Assembler pro PC. Vyd. 1. Brno: Computer Press, 2003, 228 s. ISBN 80-722-6843-0.
11. Ivan ZELINKA. Počítačové viry a bezpečnost. Prezentace PPT, Ostrava, 2012.
12. HÁK, Igor. Moderní a počítačové viry. třetí. Dostupné z: http://viry.cz/download/kniha.pdf
13. ZULFIKAR, Stamm SID a MARKUS. Drive-by pharming. 2006. Dostupné z: http://research.sidstamm.com/papers/driveby-pharming.pdf
14. Česká republika. Sbírka zákonů. In: 40. Zákon trestní zákoník. 2009, 11. Dostupné z: http://business.center.cz/business/pravo/zakony/trestni-zakonik/
15. LÁTAL, Ivo. Počítačová (informační) kriminalita a úloha policisty při jejím řesení. Policista [online]. 1998, č. 3 [cit. 2014-05-06]. Dostupné z: http://www.scritub.com/limba/ceha slovaca/Potaov-informan-kriminalita-a-1513463.php
16. Computer viruses hit one million. Http://www.bbc.co.uk/ [online]. 2014, aktualizováno 6.5.2014 [cit. 2014-05-06]. Dostupné z: http://news.bbc.co.uk/2/hi/technology/7340315.stm
17. Top 5 Computer Viruses Of All Time. Http://uk.norton.com/ [online]. ©1995 - 2014 [cit. 2014-05-06]. Dostupné z: http://uk.norton.com/top-5-viruses/promo
18. Why People Create Computer Viruses?. In: Http://www.nortonantiviruscenter.com/ [online]. © 2007-2013 [cit. 2014-05-06]. Dostupné z: http://www.nortonantiviruscenter.com/security-resource-center/why-people-create-computer-viruses.html
19. WHITTY. Why do People Create Computer Viruses?. In: Http://www.technibble.com/ [online]. © 2006-2014 [cit. 2014-05-06]. Dostupné z: http://www.technibble.com/why-do-people-create computer-viruses/
20. Slovník pojmů. In: Http://www-old.gts.cz/gtsbezpecnyinternet/cs/ [online]. © 2013 [cit. 2014-05-06]. Dostupné z: http://www- old.gts.cz/gtsbezpecnyinternet/cs/napoveda/slovnik-pojmu/
21. Speedguide.net [online]. © 1998-2014 [cit. 2014-05-06]. Dostupné z: http://www.speedguide.net/ports.php
22. List of TCP and UDP port numbers. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-2014, 6.5.2014 [cit. 2014-05-06]. Dostupné z: http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers
23. Backdoor found in D-Link router firmware code. In: InfoWorld [online]. Palo Alto, CA: InfoWorld, 1980- [cit. 2014-05-06]. Dostupné z:

http://www.infoworld.com/d/security/backdoor-found-in-d-link-router-firmware- code-228725
24. KAMKAR. NAT Pinning: Penetrating routers and firewalls from a web page (forcing router to port forward) [online]. 2010 [cit. 6.5.2014]. Dostupné z: http://samy.pl/natpin/
25. JonDonym: IP check [online]. [cit. 6.5.2014]. Dostupné z: http://ip-check.info/?lang=en
26. anopticlick. Panopticlick [online]. 2010 [cit. 2014-05-07]. Dostupné z: https://panopticlick.eff.org/
27. Check Point Survey Reveals Nearly Half of Enterprises Are Victims of Social Engineering: Social engineering attacks can cost businesses more than $100,000 per incident, emphasizing the importance of better security and user awareness. Http://www.checkpoint.com/ [online]. ©2014 [cit. 2014-05-07]. Dostupné z: http://www.checkpoint.com/press/2011/092111-enterprises-victims social- engineering.html
28. VISUAL STUDIO .NET 2003. Programming Languages [online]. [cit. 6.5.2014]. Dostupné z: http://msdn.microsoft.com/en- us/library/aa292164%28v=vs.71%29.aspx
29. Windows API [online]. [cit. 6.5.2014]. Dostupné z: http://msdn.microsoft.com/en-us/library/ff818516%28v=vs.85%29.aspx
30. PARKER, Don. Windows NTFS Alternate Data Streams. In: Symantec [online]. ©1995 - 2014 [cit. 2014-05-07]. Dostupné z: http://www.symantec.com/connect/articles/windows-ntfs-alternate-data-streams
31. The Exploit Database. Exploit Database [online]. 2014, 2.5.2014 [cit. 2014-05- 07]. Dostupné z: http://www.exploit-db.com/
32. Inj3ct0r. 1337day [online]. © 2008-2014 [cit. 2014-05-07]. Dostupné z: http://1337day.com/
33. Realtime Web Analytics With no Sampling: Desktop Operating System Market Share. NETMARKETSHARE [online]. © 2006-2014 [cit. 2014-05-07]. Dostupné z: http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0&qpsp=182&qpnp=1&qptimeframe=M
34. Windows Sockets 2. In: Windows [online]. © 2014 [cit. 2014-05-07]. Dostupné z: http://msdn.microsoft.com/en- us/library/windows/desktop/ms740673%28v=vs.85%29.aspx
35. Hacking bez záhad. McClure Stuart. Scambray, Joel. Kurtz, George. 5. aktualizované vydání. Grada 2006. ISBN: 978-80-247-1502-5
36. Hacking Manuál hackera. Harper, Allen. Harris, Shon. Eagle, Chris. Ness, Johathan. Lester, Michael. Grada 2007. ISBN: 978-80-247-1346-5.
37. Hacking Buffer Overflow zneužití a prevence. Foster C. James. Grada 2007. ISBN 978-80-247-1480-6.
38. Google Hacking. Long Johny. Zoner Press. ISBN 80-86815-22-6.
39. Hacking detekce a prevece počítačového útoku. Endorf, Carl. Schultz, Eugene. Mellander, Jim. Grada 2004. 80-247-1035-8.
40. Hacking Umění Exploitace. Erickson, Jon. Zoner Press. 2005. ISBN 80-86815-21-8
41. Velký průvodce protokoly TCP/IP a systémem DNS. Dostálek, Libor. Kabelová, Alena. Computer Press 2000. ISBN 80-7226-323-4
42. Zpovědi mladých hackerů. Verton, Dan. Helin 2003. ISBN 80-7361-243-2
43. Hacker: podivný život a zločiny počítačového génia Kevina Poulsena. Littman, Jonathan. Práh 1998. ISBN 80-85809-97