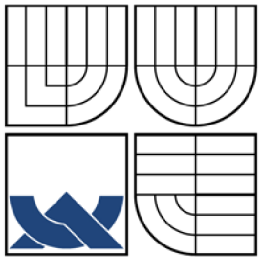
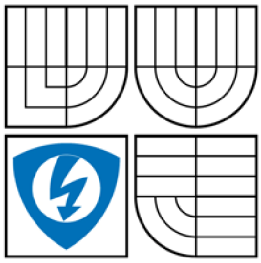


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SÍŤOVÉ ÚTOKY NA OPERAČNÍ SYSTÉM LINUX

Network attacks of Linux operating system

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE **RICHARD VALČÁK**
AUTHOR

VEDOUCÍ PRÁCE **ING. TOMÁŠ PELKA**
SUPERVISOR

BRNO 2008

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Richard Valčák

Bytem: Hodžova 3292/3, 05801 Poprad

Narozen/a (datum a místo): 23.4.1985 v Poprade

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

prof. Ing. Kamil Vrba, CSc

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - □ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP:	
Vedoucí/ školitel VŠKP:	
Ústav:	telekomunikací
Datum obhajoby VŠKP:	

VŠKP odevzdal autor nabyvateli v*:

- □ tištěné formě – počet exemplářů ……..
- □ elektronické formě – počet exemplářů ……..

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - □ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: ……

…….. ……

Nabyvatel Autor

PROHLÁŠENÍ

Prohlašuji, že bakalářskou práci na téma „**Síťové útoky na operační systém Linux**“ jsem vypracoval samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Tomáši Pelkovi, za velmi užitečnou metodickou pomoc a cenné názory při zpracovaní bakalářské práce

V Brně dne ....

(podpis autora)

ANOTACE

Cílem této bakalářské práce je seznámit se s různými nejznámějšími bezpečnostními útoky, podrobně popsat jejich vlastnosti, zhodnotit míru nebezpečnosti a navrhnout ochranu proti těmto útokům. Bakalářská práce prezentuje teoretickou a praktickou část zaměřenou na bezpečnostní útoky operačního systému Linux. V prvních kapitolách se popisuje teorie bezpečnostních vrstev operačního systému. Následují nejčastěji využívané útoky jako Buffer overlow, ARP Cache poisoning, SYN flood, Denial of Service, DHCP, které jsou teoreticky popsané, předvedené na příkladech a zjištění možné ochrany před těmito útoky je zaznamenané. Základní teoretické poznatky o systému Linux jsou potřebné pro praktickou část bakalářské práce.

Kľúčové slová: sieťové útoky, bezpečnosť, Linux, ochrana, operačný systém

ABSTRACT

Main objective of this bachelor thesis is to inform with the most famous security attacks, describe their characteristics, evaluate risk degree and suggest security against of these attacks. The bachelor work presents theoretical and practical part focused on security attacks of Linux operational system.

First chapters describe operational system security shell theory. Followed by the most frequent attacks like Buffer overflow, ARP Cache poisoning, SYN flood , Denial of Service, DHCP, which are described theoretically, demonstrated by these examples and possible security assignment against this attacks registered. Basic theoretical system Linux knowledge is necessary for the practical part of the bachelor thesis.

Keywords : network attack, security, Linux, protection, operating system

Obsah

Zoznam obrázkov

Úvod

Prínos moderných informačných a komunikačných technológií pre ďalší rozvoj jednotlivcov, organizácií i celej spoločnosti je jasný a nespochybniteľný. Počítače sa stali každodennou súčasťou nahého života. Ťažko si dnes vieme predstaviť bankový systém alebo vedecký výpočet bez pomoci počítača. S tým prichádza aj problém spoľahlivosti počítačov a dôveryhodnosti dát, ktoré nám počítače predkladajú a ktoré sme im zverili. Tieto dáta/informácie v dnešnom svete znamenajú moc, peniaze, vedomosti, závisia na nich majetky aj ľudské životy. Z problematiky spoľahlivosti počítačov sa postupne vyčlenil odbor počítačovej bezpečnosti.

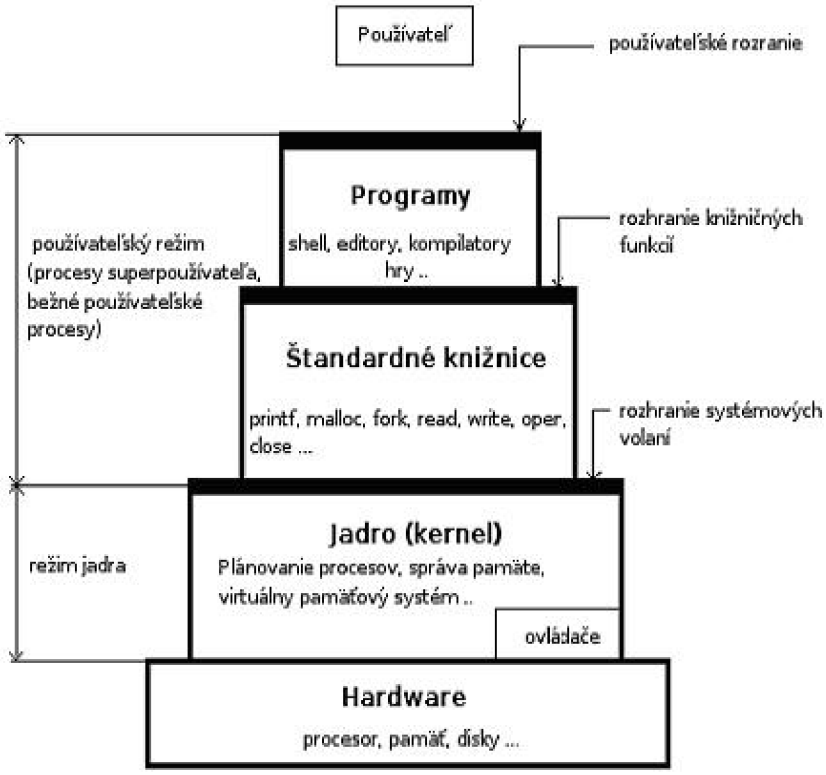
Počítačová bezpečnosť je odbor, ktorý sa zaoberá ochranou informácii pred náhodným alebo neautorizovaným zverejnením, zmenením alebo zničením. Koncepcia počítačovej bezpečnosti spočíva v troch krokoch:

- Prevencia - ochrana pred hrozbami.
- Detekcia - odhalenie neoprávnenej činnosti a slabého miesta v systéme.
- Náprava - odstránenie škôd vzniknutých pri neopravnej činnosti a slabého miesta v systéme.

V tejto práci sa budeme zaoberať hlavne prevenciou, konkrétne prevenciou pred možnosťami rozšírenia používateľských práv a pred následnou kompromitáciou operačného systému. Ako operačný systém bude použitý Linux aj keď základné princípy platia aj pre mnohé iné operačné systémy.

1. Bezpečnostné vrstvy operačného systému

Operačný systém môžeme rozdeliť na logické vrstvy na základe funkčnosti a bezpečnosti, ako to vidíme na obrázku



Obr. 1.1: Vrstvy operačného systému

1.1 Bezpečnosť na používateľskej úrovni

Na najvyššej vrstve sa nachádza používateľ, ktorý si sadá za počítač s určitým cieľom, skúsenosťami a očakávaniami. Používateľ očakáva správne nakonfigurovaný systém, ktorý sa nedostane do nekonzistentného stavu ani v prípade chyby z jeho strany. Navyše predpokladá, že program sa správa presne podľa dodaného manuálu. Správnu konfiguráciu má na starosti jeho správca. Správca inštaluje a konfiguruje programové vybavenie systému.

Prideľuje práva používateľom na prístup k dátovým súborom a programom. Správca takto vytvára bezpečnosť na používateľskej úrovni. Treba poznamenať, že práve táto vrstva je najslabším článkom v počítačovej bezpečnosti. Hlavným dôvodom je závislosť na ľudskom faktore. Správca aj používateľ mnohokrát mylne predpokladajú, že o ich dáta nemá nikto záujem. Neuvedomujú si riziko prezvania cudzej kontroly nad počítačom a podceňujú riziko prezradenia aj malo dôležitých dát.

1.2 Bezpečnosť na aplikačnej úrovni

Program sa považuje sa bezpečný, ak splna dve nasledujúce podmienky. Prvá je tzv. vonkajšie správanie, ktoré by malo zodpovedať správaniu v používateľskej príručke. Druhou podmienkou bezpečnosti programu je vnútorné správanie – vlastná implementácia. Zlá implementácia môže znamenať napr. chybné ošetrenie vstupných dát od používateľa, ktoré môže viesť k zápisu mimo pridelenú pamäť. Vtedy môže dôjsť k mimoriadne nebezpečným situáciám ako k spusteniu externého kódu alebo pádu systému. Z hľadiska bezpečnosti programu je najčastejšou chybou pri programovaní zlé ošetrenie vstupu od používateľa. Na druhom mieste je používanie knižničných funkcií, ktoré sú z určitého hľadiska nevhodné. Poslednou časťou chybou je zabudnutie odstránenia testovacích procedúr z vývojovej fázy programu. Linux je viacužívateľský operačný systém a preto je nutné zabezpečiť, aby si používatelia nemohli navzájom prepisovať súbory a spúšťať programy, ktoré im nepatria.

Technológie riadenia prístupu sa časom zdokonaľovali, pričom sa vyčlenili dva hlavné typy: *voliteľné riadenie prístupu (DAC – Discretionary Access Control)* [6] a *povinné riadenie prístupu (MAC – Mandatory Access Control)*. DAC dáva možnosť jednotlivým používateľom prideľovať a odoberať prístupové práva. V tejto schéme môžu byť používatelia taktiež vlastníkmi objektov. Systémy založené na DAC dovoľujú používateľom povoliť alebo zakázať prístup k ľubovoľným objektom, ktoré vlastnia. Problém môže nastať, ak používateľ nie je skutočným majiteľom informácie, ktorej vlastníkom je v DAC systéme (príkladom môže byť lekár, ktorý má právo čítať lekárske záznamy pacienta, ale nie je ich skutočným majiteľom a teda nemá právo ich ďalej poskytnúť tretej strane – toto právo má len pacient). Často skutočným majiteľom informácií (objektov), ako aj programov, ktoré tieto informácie spracúvajú, je samotná organizácia.

Preto je v niektorých prípadoch výhodnejší použiť iný typ riadenia prístupu napr. MAC, kde sú práva prístupu riadené organizáciou.

MAC je spôsob riadenia prístupu k objektom založený na stupni dôvernosti informácie v nich obsiahnutej (stupeň dôvernosti určuje organizácia a nie vlastník informácie) a autorizácii používateľa pristupovať k informáciám označeným určitým stupňom dôvernosti To znamená, že organizácia určí kto môže pristupovať ku informáciám určitého stupňa dôvernosti. Riadenie prístupu je povinné v zmysle, že používateľ nemá právo meniť stupeň dôvernosti informácie a tým ju poskytnúť používateľom s menšími právami.

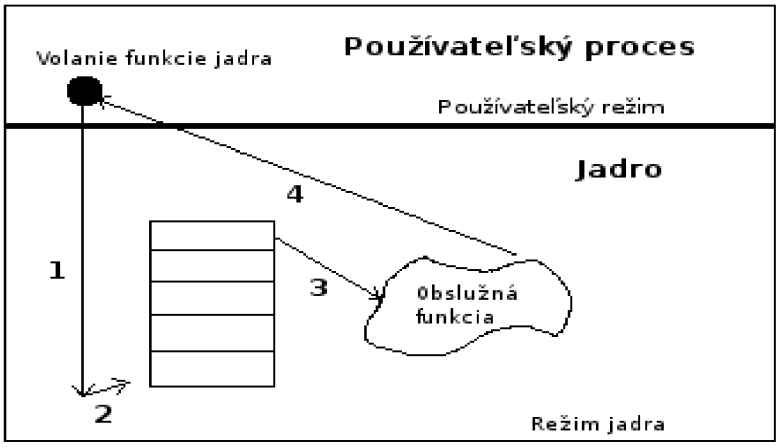
1.3 Bezpečnosť na úrovni jadra

Bezpečnosť na úrovni jadra súvisí priamo so stabilitou celého systému. Chyba na tejto úrovni môže spôsobiť pád spustených procesov, dôsledkom čoho môže dôjsť k najzávažnejším chybám, ako je napríklad poškodenie súborového systému. Jadro je zodpovedné za vytvorenie prostredia pre beh ostatných používateľských programov. To znamená, že pre každý proces jadro vykonáva alokáciu a ochranu pamäťového priestoru, umožňuje prístup k periférnym zariadeniam. Na pozadí tvorby tohoto prostredia jadro komunikuje s ovládačmi jednotlivých zariadení, riadi stránkovanie virtuálnej pamäte a prevádza kontrolu oprávnenia požadovaných akcií jednotlivými procesmi.

2. Operačný systém s unixovým jadrom

V Unixových systémoch je snaha o oddelenie používateľských procesov od procesov jadra z dôvodov zachovania väčšej stability a bezpečnosti jadra. Preto používateľské procesy bežia v tzv. používateľskom režime, jadro beží v režime jadra. Oba tieto režimy majú oddelené kódové a dátové segmenty. V používateľskom režime procesor automaticky obmeňuje používanie privilegovaných inštrukcií ako napr. prístup k I/O portom alebo nastavenie stránkovania pamäte. Aby mal používateľ možnosť napr. citát z disku, čo je I/O operácia, musí jeho program vykonať systémové volanie, ktoré jadro poskytuje. Pamäť každého programu je rozdelená na päť časti: text, data, bss (block started by symbol, časť

neinicializovaných dát), heap(halda) a stack (zásobník). Každá časť reprezentuje špeciálny druh pamäte s rozdielnymi vlastnosťami.



Obr. 2.1: Volanie funkcií jadra

Časť text sa niekedy označuje ako code. Je to miesto, kde sa nachádzajú inštrukcie strojového jazyka. Vykonávanie inštrukcii v tejto časti nie je lineárne, pretože sa používajú inštrukcie vetvenia (branch), skokov a volaní. Pri vykonávaní kódu procesor využíva svoju relatívne malú pamäť - registre. Časť z týchto registrov je vyhradená na udržiavanie informácií o momentálne spustenom programe. Jeden z najdôležitejších je EIP (extended instruction pointer), ktorý ukazuje na práve vykonávanú inštrukciu. V okamžiku spustenia programu sa EIP nastaví na prvú inštrukciu časti text.

Procesor potom robí následný cyklus:

1. Prečítaj inštrukciu, na ktorú ukazuje EIP,
2. pripočítaj k EIP dĺžku inštrukcie,
3. vykonaj inštrukciu prečítanú v kroku 1,
4. choď na krok 1.

Niekedy je prečítaná inštrukcia inštrukciou skoku alebo volanie, ktorá mení EIP na inú adresu. Procesor sa nestará o zmeny, predpokladá nelineárne vykonávanie. Takže, keď

sa v kroku 3 EIP nejako zmení, procesor bude ďalej pokračovať v kroku 1, prečíta a vykoná ďalšiu inštrukciu, nech je akákoľvek.

Právo k zápisu je v časti text vypnuté, pretože neslúži k uchovávaniu premenných ale kódu. To zabraňuje modifikovaniu programového kódu počas behu programu a každý pokus o zápis do tejto časti pamäte konci okamžitým ukončením programu. Ďalšia výhoda, ktorá z toho plynie je možnosť zdieľania časti text pri spustení viac kópii toho istého programu. Táto časť pamäte má fixnú veľkosť. Časti dáta a bss sa používajú pre uchovávanie globálnych a statických premenných. V časti dáta sa nachádzajú inicializované globálne premenné, reťazce a konštanty, ktoré sa v programe používajú. V bss sú neinicializované premenné.

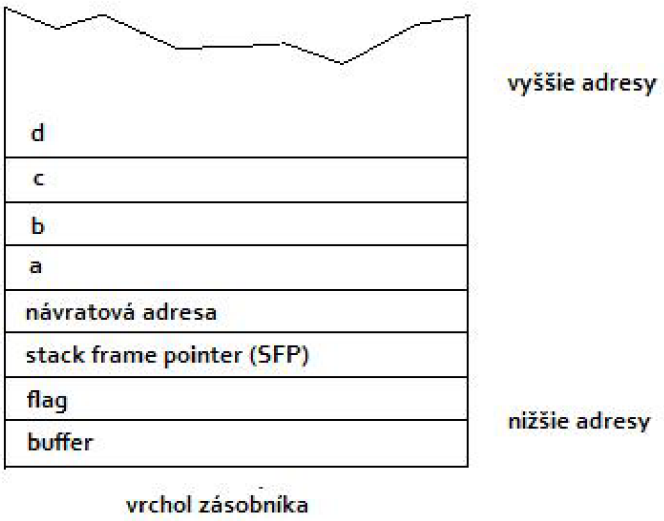
Napriek tomu, že sú tieto časti zapisovateľné, majú fixnú veľkosť. Časť heap sa používa na uloženie ostatných premenných. Jej veľkosť rastie a klesá podľa potreby. Celá táto pamäť je riadená alokačnými a dealokačnými algoritmami jadra, ktoré rezervujú časť pamäte pre neskoršie použitie a spätne odstraňujú rezerváciu, aby mohla byť táto pamäť využitá. Pamäťová časť programu stack má tiež premennú veľkosť a používa sa ako dočasné úložisko pre kontext funkcie (všetky premenné, ktoré je potrebné si zapamätať) pri volaní. Pretože sa kontext musí pri volaní funkcie zmeniť, zásobník slúži k zapamätaniu si všetkých týchto premenných vrátane EIP, na ktorý sa po skončení funkcie program vráti.

Veľkosť zásobníka rastie z vyšších adries k nižším. Keď sa zavolá funkcia, potrebné údaje sa vložia na zásobník vo forme tzv. rámca zásobníka (stack frame). Register EBP, niekedy nazývaný ako frame pointer (FP, ukazovateľ na rámec) alebo local base pointer (LB, ukazovateľ na lokálnu bázu) sa použije k odkazovaniu na premenné v aktuálnom zásobníkovom rámci. Každý taký rámec obsahuje parametre funkcií, jej lokálne premenné a dva pointery dôležité pre návrat do stavu, aký bol pred volaním: saved frame pointer (SFP, uložený ukazovateľ na rámec) a return value (návratová adresa). Ukazovateľ na rámec zásobníka sa používa k obnoveniu EBP na jeho predchádzajúcu hodnotu a návratová adresa k obnoveniu EIP na ďalšiu inštrukciu hneď za volaním funkcie.

```
viod test(int a, int b, int c, int d) {
char flag;
char buffer[10];
}
void main() {
test(1,2,3,4);
}
```

Vo vyššie uvedenej ukážke deklarujeme funkciu test so štyrmi číselnými argumentmi a, b, c, d. Medzi lokálne premenné funkcie patrí jeden znak flag a desať znakové pole nazvané *buffer*. Funkcia main() sa po spustení programu spustí ako prvá a jednoducho zavolá funkciu test(). Keď sa funkcia test() zavolá z funkcie main(), uložia sa na zásobník rozličné hodnoty a vytvorí sa rámec zásobníka. Argumenty funkcie sa ukladajú najskôr a to v opačnom poradí, teda d, c, b a nakoniec a. V momente, keď je spustená inštrukcia call pre volanie funkcie test(), sa na zásobník pridá návratová adresa, ktorej hodnota bude hodnota EIP zväčšená o inštrukciu volania (teda adresa inštrukcie call + veľkosť samotnej inštrukcie).

V ďalšom kroku sa na zásobník uloží hodnota registra EBP, čo je uložený ukazovateľ na rámec a neskôr sa použije na obnovenie pôvodného stavu. Aktuálna hodnota registra ESP sa potom skopíruje do registra EBP a tým sa nastaví ukazovateľ na nový rámec. Nakoniec sa alokuje pamäť pre lokálne premenné funkcie (flag a buffer) na zásobníku zmenšením ESP. Na konci to vyzerá ako na obr.2.2. Týchto päť časti pamäte programu je za sebou poukladaných v poradí, v akom boli uvedené, od nižších adries po vyššie. Keďže sú časti heap a stack dynamické rastú oproti sebe, tým sa minimalizuje možnosť plytvania miestom. Poslednou časťou sú oblasti, ktoré vznikajú pri zavádzaní programu do pamäte podľa predpisu spustitelného formátu. Spustitelný formát obsahuje sekcie, ktoré obsahujú kód programu, statické dáta, relokačnú tabuľku, informácie potrebné pri linkovaní a iné. Výpis sekcií programu je možné získať napr. príkazom readelf –e program.



Obr. 2.2: Zásobník

3. Práva

Každému súboru v Unixe môžeme nastaviť prístupové pravá (**read, write, eXecute**). Sú to owner (vlastník), group (skupina) a others (ostatný). Nastavenie práv pre každú z týchto množín je úplne nezávislé. Množiny sa označujú znakmi **u, g a o** a pravá prístupu **r, w a x**. Ešte poznamenajme, že pravo spúšťania pre adresár znamená, že do adresára môžeme vojsť. Každý súbor môže mať pravá typu: `-rwxrwxrwx` (prvé tri písmena definujú pravá pre *ownera* ,stredne pre *group* a posledne pre *others*. Berieme to zľava doprava. Predstavte si binárny kód... miesto najviac vpravo je **1**, miesto v strede je **2** a miesto celkom vľavo je **4** - pre každú množinu berieme jednu číslicu**.** Čiže pravá na čítanie pre všetky skupiny (`r--r--r--`) nie je nič iné, ako čísla 4, 4 a ešte raz 4. Čiže píšeme chmod 444 súbor. Takisto pravá (`rw-r--r--`) môžeme napišať číselné - bude to 644...

Tu je niekoľko vysvetlených príkladov:

`chmod ugo=r subor`
`chmod 444 subor`
`chmod u=r,g=r,o=r subor`
	nastavíme pravá súboru subor na čítanie pre,
	všetky množiny užívateľov,

`chmod ugo=rx *`	nastavíme pravá všetkých súborov v adresári na čítanie a spúšťanie pre všetky množiny užívateľov,
`chmod ugo=rx,u+w *`	nastavíme pravá všetkých súborov v adresári na čítanie a spúšťanie pre **group a others** + zapisovanie pre **owner,**
`chmod o-r *`	odoberieme právo na čítanie všetkých súborov v adresári **others,**
`chmod o=r,w=wx fajl`	nastavíme pravá súboru **fajl** pre **others** na čítanie a sebe povolíme zapisovanie a spúšťanie.

3.1 i-node

Základnou jednotkou súborových systémov používaných v operačných systémoch typu UNIX, teda vzťahuje sa to aj na Linux, je takzvaný i-node. i-node je zvláštna riadiaca štruktúra, ktorá ukazuje na ďalšie i-nody alebo na bloky dát. Obsahuje údaje o majiteľovi súboru, právach, veľkosti, čase posledného prístupu k nemu, čase jeho vytvorenia atď. Čo však v i-node uložené nie je, je názov súboru. Adresár je tiež špeciálnym typom súboru. To znamená, že pre každý adresár existuje i-node alebo niekoľko i-nodov, ktoré ukazujú na bloky dát, v ktorých sú zapísané údaje, teda názvy a i-nody, týkajúce sa súborov v adresári.
i-nody sa používajú aj na vytváranie nepriamych odkazov, aby bolo možné odkazovať sa na väčšie množstvo blokov dát. Práve preto i-node neobsahuje názov súboru. Jeden súbor reprezentuje vždy len jeden i-node a ukladanie názvov súborov do všetkých i-nodov sa považovalo pri vzniku unixových systémov za plytvanie miestom. Ak by totiž každý i-node niesol so sebou aj názov súboru, musel by sa zväčšiť o 255 bajtov potrebných k uloženiu súboru.

i-node obsahuje:

- režim súboru: práva na čítanie, zápis, vykonanie,
- ID vlastníka,
- veľkosť súboru v bajtoch,
- čas posledného prístupu (atime),
- Čas poslednej zmeny inode (ctime),
- Čas poslednej zmeny súboru (mtime),
- Čas zmazania (dtime),
- ID skupiny,
- počet odkazov,
- počet blokov,
- flags,
- použité bloky ,
    - priame bloky,
    - nepriame bloky,
    - dvojnásobne nepriame bloky.

4. LIDS (Linux Intrusion Detection System)

LIDS (Linux Intrusion Detection System (Linuxový systém detekcie prieniku)) je vylepšenie pre linux, ktoré naprogramovali Xie Huagang a Philippe Biondi. Implementuje viacero bezpečnostných funkcií, ktoré neboli až do doby SELinuxu v štandardnom Linuxovom jadre (momentálne sa v jadre nachádza bezpečnostný modul od NSA z názvom SeLinux).

Vlastnosti

- Ochrana súborov a adresárov. Nikto vrátane administrátora (root) nemôže modifikovať súbory chránené pomocou LIDS. Súbory a adresáre môžu byť aj neviditeľné.
- Ochrana procesov. Nikto vrátane administrátora (root) nemôže ukončiť pomocou signálu kill chránené procesy. Procesy môžu být aj neviditeľné.

- *Access Control Lists (ACLs)* pre prístupové práva k súborom, adresárom (*File ACLs*) a *ACLs*, ktoré obmedzujú schopnosti a systémové možnosti procesov (*Capability ACLs*).
- Rozšírená schopnosť kontrolovať celý systém.
- Bezpečnostné upozornenia od jadra operačného systému. SMTP klient sa voliteľne môže stať súčasťou jadra.
- IDS - systém detekcie prieniku.

5. Najčastejšie spôsoby narušenia bezpečnosti

5.1 Buffer overflow

Buffer overflow - zápis mimo rozsah alokovanej pamäte.

Väčšina imperatívnych programovacích jazykov dovoľuje alokovat pamäť pre uloženie premenných a dát viacerými spôsobmi. Rozpoznávame dva druhy alokácie pamäte, statickú a dynamickú. Pri statickej je veľkosť alokovanej pamäte známa už pri preklade a počas behu programu už nie je možné túto veľkosť meniť. Dynamická alokácia má práve tú vlastnosť, že je možné meniť jej veľkosť počas behu programu v závislosti od požiadaviek. Oba druhy alokácie sa líšia syntaxou zápisu a umiestnením alokovanej pamäte vo virtuálnom adresnom priestore procesu.

Dynamická alokácia - Dynamickú alokáciu môžeme v jazyku C uskutočniť pomocou štyroch funkcií. Dynamicky alokované miesto sa nachádza v oblasti haldy. Oblasť haldy vzniká pri prvom použití funkcie malloc() (alebo podobných) zväčšením dátovej casti programu. Volanie jadra brk() sa stará o zväčšenie dátovej časti procesu. Vždy pri blokovaní pamäte sa vytvorí štruktúra označovaná ako úsek pamäte (po anglicky memory chunk), ktorá umožňuje neskoršie uvoľnenie alebo zlučovanie oblastí.

```
void * calloc(int members, int size);
void * malloc(int size);
void * realloc(void *ptr, int size);
void free(void *ptr);
```

Statická alokácia - Staticky alokované oblasti sa môžu nachádzať buď v dátovej oblasti alebo na zásobníku. O alokovanie a uvoľňovanie pamäte sa stará zavádzač programu. Vrchol zásobníka inicializuje na hodnotu 0xbfffffff, prideľovanie pamäťových stránok

zásobníku dynamicky obstaráva jadro. Ďalšou výhodou tohto spôsobu alokácie je to, že je o mnoho rýchlejšia ako dynamická. char pole;

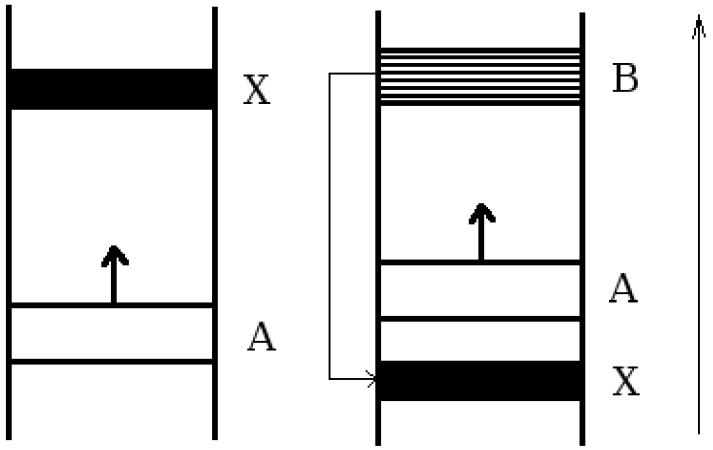
Z dôvodu rýchlosti a efektívnosti výsledného programu kompilátory jazyka C nerobia žiadnu kontrolu zápisu mimo hranice alokovanej pamäte, všetko toto nechávajú na programátora. V chybne napísanom programe môže dôjsť k zápisu mimo hranice alokovanej pamäte a k následnému prepísaniu hodnôt premenných a tým k pádu programu alebo k spusteniu externého programu. Anglicky sa táto situácia označuje buffer overflow.

```
program: buffer.c
#include<stdio.h>
int main(int argc, char *argv[]) {
int i = 0,a = 1;
int buffer[3];
printf ("a: %d\n",a);
for (i = 0;i <= 3;i++)
buffer[i] = 2;
printf ("a: %d\n",a);
return 0;
}

Kompilácia a spustenie:
$gcc buffer.c -o buffer
$./buffer
a: 1
a: 2
$
```

Po spustení funkcie main() dochádza k jej prologu, tj. uloženiu registra EBP a nastaveniu EBP na novú hodnotu, ktorá určuje lokálny rámec funkcie main(). Alokuje sa miesto pre lokálne premenné a dochádza k postupnému ukladaniu hodnoty 2 na adresy (buffer + i). Adresa (buffer + 3) však patrí premennej a, takže dôjde k jej prepísaniu. Cieľom útočníka je väčšinou prepísanie návratovej adresy funkcie, týmto spôsobom môže zmeniť tok programu a využiť to pre svoj prospech a ciel.

Druhy zápisov mimo alokovanú pamäť Podľa toho, v ktorej pamäťovej oblasti došlo k zápisu mimo vyhradenú pamäť, anglická literatúra používa termíny stack, heap, bss a data overflow. Prakticky oblasť, v ktorej k pretečeniu došlo, určuje možnosti útoku. Vo virtuálnom adresnom priestore existuje niekoľko adries, ktorých prepísanie môže viesť k spusteniu externe vloženého kódu. Okrem spúšťania externého kódu môže dochádzať ku poškodzovaniu dát. Metódy zápisu mimo medze sa dajú rozdeliť na priame a nepriame.



Obr. 5.1: Priamy a nepriamy zápis

Nech A je alokovaná oblasť, pri ktorej môže dojsť zápisu mimo medze, B je ukazovateľ na A, X je cieľová adresa zápisu. Priamy zápis je jednoduchý a videli sme ho v programe buffer.c. Nevýhodou je možnosť umiestnenia cieľových adries iba na vyšších adresách vzhľadom k A. Tiež musí byt splnená podmienka, že prepísanie dát medzi A a X nespôsobí pád programu. Pri nepriamom zápise dochádza k prepísaniu hodnoty ukazovateľa B, tak aby ukazoval na X. Ak sa v programe vyskytuje nasledovné kopírovanie dát na adresu B, potom dochádza k zápisu na cieľovú adresu X. Výhodou tohto spôsobu je, že takto je možné získať prístup na ľubovoľnú adresu, nevýhodou je väčšia zložitosť.

5.1.1 Prepísanie návratovej adresy funkcie

Pomocou kopírovania väčšieho množstva dát, ako je očakávané, dokážeme prepísať premenné na vyšších pamäťových miestach tj. Nižšie v zásobníku. V tejto časti sa budeme zaoberať práve prepísaním návratovej adresy funkcie, čo je jeden z najjednoduchších a zároveň najpoužívanejších spôsobov zneužitia programov. Obľúbenosť tejto metódy je založená na jej efektivite, pretože ak sa nám podarí prepísať návratovú adresu setuid programu na náš kód, ktorý napríklad spustí shell, tak sme sa práve stali superpoužívatelom s neobmedzenými právami. Tento náš kód, ktorý spustí shell sa volá shellkód, je to

postupnosť bajtov, ktorá je interpretovaná ako strojový kód. Samozrejme, bolo by možné priamo napišať strojový kód na prevedenie konkrétnej požadovanej operácie, ale to by bolo zbytočné komplikované. Existuje veľké množstvo druhov shellkódu, ktoré sú závislé na rôznych prostrediach (architektúra, operačný systém). Vytvorenie shellkódu bez znalostí jazyka Assembler je možné pomocou knižnice libShellCode1. Je to open-source knižnica, ktorá umožňuje vytvorenie shellkódu s potrebnými charakteristikami za behu pre systémy Linux a BSD. Ked už máme vytvorený shellkód, musíme vyriešiť otázku jeho umiestnenia v adresnom priestore programu. Ak má program dostatočné veľký vstupný buffer na umiestnenie celého shellkódu, môžeme ho zadať ako vstup. Ak nie, môžeme využiť premenné prostredia (enviroment) na uloženie shellkódu a potom presmerovať vykonávanie programu na túto adresu.

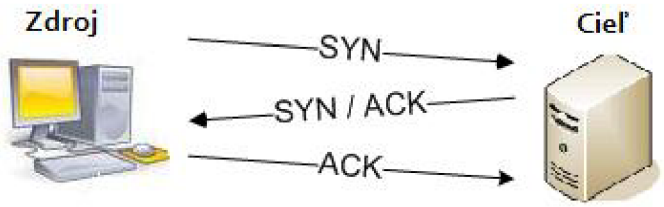
5.2 SYN flood

Útok typu syn flood (záplava SYN paketov) má za úlohu vyčerpať kapacitu obeti. TCP/IP siete boli navrhnuté v otvorenom a dôveryhodnom prostredí, čo sa na ich bezpečnosti negatívne podpísalo. Dôveryhodnosť TCP/IP zneužíva aj útok zaplavovanie SYN paketov. Útok využíval zle implementácie, nikto presne s týmto útokom nepočítal. Výmena dát cez TCP/IP začína „trojitým potrasením rúk“ (three-way handshake). Toto potrasenie za bežných okolností začína SYN paketom, ktorý pošle počítač na nejaký otvorený (listen) port počítača B. Tento port prejde do stavu SYN_RECV a počítač B počítaču A odpovie paketom s príznakom SYN/ACK. Nakoniec pošle počítač A počítaču B paket ACK a tým je spojenie naviazane (established).

Normálna TCP komunikácia sa skladá z troch krokov (trojfázový handshake):

1. zdrojový počítač vyšle TCP paket s nastaveným príznakom "**SYN**" (žiadosť o komunikáciu),
2. cieľový počítač odpovedá paketom s nastavenými príznakmi "**SYN**" a "**ACK**" (pripravený na komunikáciu),
3. zdrojový počítač odpovedá paketom s nastaveným príznakom "**ACK**" (začíname komunikovať).

Po tejto výmene je spojenie nadviazané a potvrdené a nasleduje normálna dátová komunikácia.



Obr. 5.2: SYN flood

Na spracovaní napol otvorených spojení v tvare SYN_RECV väčšina systémov vyhradzuje len obmedzený počet prostriedkov. Ale už niekoľko desiatok napol otvorených spojení úplne znemožní ďalšiu sieťovú komunikáciu. Presne v tom spočíva podstata SYN útoku.

Pri SYN útoku pošle hacker obeti SYN paket s falošnou zdrojovou adresou. Obeť na túto adresu odpovie SYN/ACK paketom, na čo nič netušiaci cieľový systém (pokiaľ vôbec existuje) reaguje RST paketom. Cieľový systém ale väčšinou neexistuje, takže obeť sa RST paketu ani dokončenia spojenia nedočká a port zostane v stave SYN_RECV. V tomto stave port zostane, pokiaľ nevyprší maximálna povolená doba naviazania pripojenia, ktorá je nastavená podľa konkrétneho systému. Rada určená pre napol otvorené porty má väčšinou len malú kapacitu, takže útočníkovi stačí poslať SYN paket napríklad raz za desať sekúnd a cieľový port bude k ničomu, pretože ešte pred jeho navrátením medzi volne porty príde ďalší SYN paket a napol otvorený port opäť skončí v fronte. Pokiaľ takých požiadavkou príde veľké množstvo, server má otvorené veľké množstvo spojení a nieje schopný obslúžiť ďalších užívateľov. V prípade veľkého množstva falošných požiadaviek potom len vlastne čaká na odpoveď a neobsluhuje nikoho, pretože nieje schopný otvoriť ďalšie spojenie. Proti útokom typu syn flood neexistuje v podstate žiadna ochrana, pretože využívajú spôsob definovaného pre zahájenie TCP relácie.

Útok SYN pakety je nepríjemný pretože útočník si vystačí veľmi skromným pripojením k sieti a používa falošnú zdrojovú adresu paketu, takže je ťažké určiť skutočného pôvodcu útoku.

5.2.1 Prax

Z nižšie uvedenej sieťovej komunikácie je patrné, ako šifrovať SYN paketom zahajuje komunikáciu s komunikačnou jednotkou.

Source address	Destination address	Protocol	Info
10.10.12.12	10.10.11.10	TCP	1105 > 2000 [SYN]
10.10.11.10	10.10.12.12	TCP	2000 >1105[SYN,ACK]
10.10.12.12	10.10.11.10	TCP	1105 > 2000 [ACK]

Syn flood útok spustíme pomocou programu hping na IP adresu a port zistený pri skenovaní siete: `hping2 -S -flood - rand-source 10.10.11.10 -p 2000`. Takto zahltíme port 2000 SYN pakety ako vidieť na nižšie uvedenej sieťovej komunikácií.

Source address	Destination address	Protocol	Info
174.106.199.13	10.10.11.10	TCP	2204 > 2000 [SYN]
10.10.11.10	174.106.199.13	TCP	2000 > 2204 [RST, ACK]
68.82.171.246	10.10.11.10	TCP	2205 > 2000 [SYN]
10.10.11.10	68.82.171.246	TCP	2000 > 2205 [RST, ACK]

Použitím SYN-cookies dovoľuje serveru znížiť počet zahodených žiadosti o spojenie pri naplnení fronty pripravovaných spojení, čo práve je cieľom útočníka. Server sa chová tak, akoby sa fronta zväčšila. Pokiaľ server obdrží žiadosť o spojenie (paket s príznakom SYN), pošle klientovi kladnú odpoveď, teda paket s príznakmi SYN a ACK. Avšak na rozdiel od obvyklého spôsobu, žiadosť potom vyhodí z fronty. A keď príde od klienta paket s príznakom ACK, server podľa čísla sekvencie pozná, o ktorú žiadosť sa jednalo. Tým je problém so zaplavením žiadosťami o spojenie vyriešený, pretože server si v pamäti nič neukladá.

SYN-cookies sú počiatočné čísla sekvencií **n**, ktoré sú poctivo vyberané podľa nasledujúcich pravidiel.

- Nech **t** je čítač zvyšujúci sa každých 64 sekúnd.
- Nech **m** je maximálna veľkosť segmentu, ktorou by mal server uchovávať v jednom zázname fronty SYN.
- Nech **s** je 24-bitový výsledok tajnej , kryptografickej funkcie počítaný cez IP adresu servera aj klienta, číslo portu a hodnotu čítača **t**.
- To znamená, že počiatočná TCP sekvencia alebo SYN – cookie sa počíta napríklad takto:

- Prvých 5 bitov: **t** mod 32.
- Ďalšie 3 bity: zakódovaná hodnota reprezentujúca číslo **m.**
- Ostatných 24 bitů: **s.**

(Pozn.: Pretože m sa kóduje na tri bity, server môže použiť len 8 jedinečných hodnôt **m**.)

Server zvolil počiatočné číslo TCP - sekvencie a klient je povinný podľa špecifikácie protokolu TCP toto rešpektovať. Paket s príznakmi SYN a ACT príde od klienta s číslom sekvencie n+1. Potom urobí tieto operácie:

- Skontroluje hodnotu **t**, aby zistil či už nevypršal časový limit.
- Prepočíta **s**, aby zistil či je SYN – cookie platná .
- Zistí hodnotu m z 3-bitového kódu v SYN-cookie, čo mu pomôže rekonštruovať záznam v fronte SYN.

Pokiaľ je všetko v poriadku, pokračuje sa v naviazaní podľa obvyklého scenára: pošle TCP paket s príkazom ACK. Zo sieťovej komunikácie vidíme, že hping používa falošnú zdrojovú adresu, takže je ťažké určiť skutočného pôvodcu útoku. Komunikačná jednotka odpovedá na záplavu SYN paketov reset paketom. SYN paketov príde veľké množstvo a tým ma elektromer otvorené veľké množstvo spojení a nieje schopný otvoriť ďalšie spojenie. Po tomto útoku sa nemôže šifrátor spojiť s komunikačnou jednotkou a to znamená, že nemôže odčítať dáta, takže útok bol úspešný. Útok SYN paketov [15] je nepríjemný, pretože si

vystačí s veľmi skromným pripojením k sieti využíva spôsob definovaného pre zahájenie TCP relácie a pracuje s veľkým objemom dát ale útok funguje jedine vtedy, keď server alokuje prostriedky po obdržaní paketu SYN ešte pred tým ako obdržal paket ACK. Základnou obranou je skrátiť dobu po ktorú server čaká na pokračovanie relácie vyvolené príkazom SYN, alebo blokovať prevádzku prichádzajúcu z falošných IP adries. Je možné riešenie i v podobe SYN a RST Cookie, ktoré odstraňuje problém s alokáciu systémových prostriedkov pred úplným naviazaním spojenia..

5.2.2 Obrana proti útokom typu syn flood

- Skrátiť dobu, po ktorú server čaká na pokračovanie relácie vyvolané príkazom syn po odpovedi ack.
- Blokovanie prevádzky, ktorá prichádza z útočných falošných IP adries.
- Používaním filtrovania paketov tečúcich z vašej siete, aby ste zabránili zneužitiu vlastných serverov pre rolu útočníka na ďalšie servery.
- Povoliť viac pootvorených spojení a znížiť dobu, po ktorú sa čaká na odpoveď.
- Riešenie v podobe SYN a RST Cookie. Riešenie Cookies odstraňuje problém s alokaciou systémových prostriedkov pred úplným naviazaním spojenia.

5.3 ARP Cache poisoning

Ide o techniku, ktorá využíva slabiny v ARP protokole. Naším hlavným cieľom je dostať dáta niekoho iného k našému počítaču, aby sme si ich mohli pozrieť.

Nachádzame sa v sieti, kde je spojovacím zariadením switch. Switch pracuje na linkovej vrstve a adresuje pomocou MAC adries. Každý switch ma vnútri pamäť do ktorej si ukladá MAC adresy zariadenia s sieti a priraďuje si k nim porty, ku ktorým sú tieto zariadenia pripojené. Keď mu potom prídu dáta, on sa pozrie pre akú MAC adresu je určená, Vyhľadá túto adresu v pamäti a pozrie sa ku ktorému portu je táto adresa priradená a dáta pošle týmto portom.

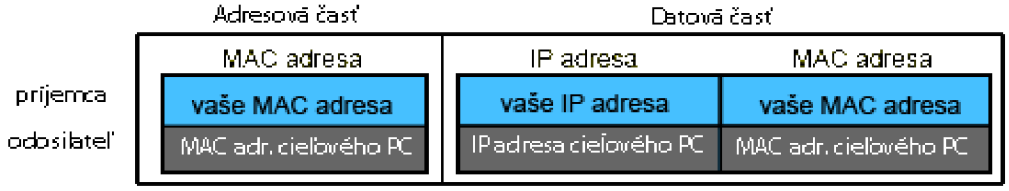
Protokol ARP slúži k prekladu IP adries na MAC adresy. Medzi IP adresou a MAC adresou nieje žiadna matematická spojitosť, preto tento preklad nemôže prebehnúť výpočtom.

Váš počítač si z paketu odpovedi vytiahne MAC adresu odosielateľa z dátovej časti. Teraz váš počítač vie akú MAC adresu má daný počítač. Aby nemusel tento preklad robiť stále urobí si záznam do pamäte, že k určitej IP adrese patrí určitá MAC adresa. Tento záznam je uložený nejakú dobu v pamäti. Potom je vymazaný a preklad prebehne znova. Týmto pamäťam hovoríme ARP Cache.

5.3.1 Teória útoku

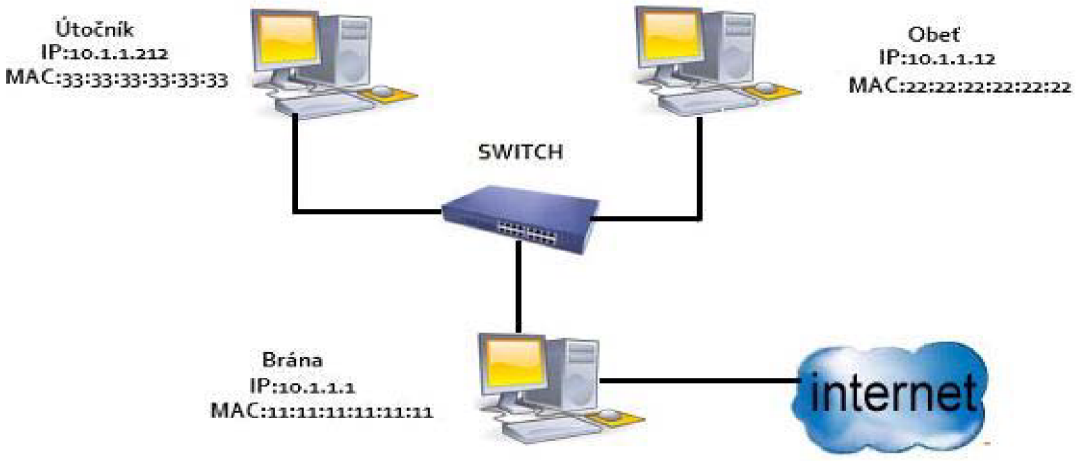
Útok na ARP presmerovania ukážem na nasledujúcom príklade :
Budeme pracovať s troma sieťovými stanicami pripojených k jednému prepínaču. Majme teda počítač s IP adresou 10.1.1.1 a menom „brána“ počítač „obeť“ s IP adresou 10.1.1.12 a počítač „útočník“ s IP adresou 10.1.1.212.

Naším cieľom bude odchytenie všetkých paketov, ktoré z počítača obeť odchádzajú cez východziu bránu preč. Brána je počítač, ku ktorému je pripojený Internet, musí komunikovať cez neho.



Obr. 5.3: Ukážka vyplnenia adries u paketu ARP Replay

Útok by sa realizoval tak, že počítače s menom obeť pošleme paket, v ktorom mu hovoríme, že vychodia brána má MAC adresu rovnakú ako útočník. Ďalej by sme poslali paket bráne, že obeť má MAC adresu rovnakú ako útočník. Tým docielime, že pre vzájomnú komunikáciu budú dosadzovať MAC adresu útočníka a switch tak posiela dáta nám. My by sme si mohli dáta pozrieť a poslať ich ďalej, ale už musíme vyplniť správnu MAC adresu.
Útok by sa nám podaril, pretože protokol ARP si vôbec nestráži, či o tieto dáta žiadal, alebo nie. Akonáhle už má záznam vytvorený, akýmkoľvek paketom ARP Reply mu môžeme záznam zmeniť. Jediná podmienka, ktorá musí byť splnená, je aby už bol záznam (priradenie MAC adresy k IP adrese), ktorý chceme zmeniť, v cieľovom počítači zmeniť. To tiež nieje problém, pretože môžeme poslať napríklad ping s falošným odosielateľom. Počítač dostane tieto dáta a sám si uloží odosielateľovu IP a MAC adresu



Obr. 5.4: Zapojenie pracoviska

ARP Cache Oběti	ARP Cache Brány
10.1.1.212 = 33:33:33:33:33:33	10.1.1.12 = 22:22:22:22:22:22
10.1.1.1 = 11:11:11:11:11:11	10.1.1.212 = 33:33:33:33:33:33

ARP Cache Oběti	ARP Cache Brány
10.1.1.212 = 33:33:33:33:33:33	10.1.1.12 = *33:33:33:33:33:33*
10.1.1.1 = 11:11:11:11:11:11	10.1.1.212 = 33:33:33:33:33:33

ARP Cache Oběti	ARP Cache Brány
10.1.1.212 = 33:33:33:33:33:33	10.1.1.12 = *33:33:33:33:33:33*
10.1.1.1 = *33:33:33:33:33:33*	10.1.1.212 = 33:33:33:33:33:33

Obr. 5.5: ARP Cache

5.3.2 Prax

Máme naše tri počítače: obeť, brána a útočník. Najprv musíme vytvoriť záznamy v ARP Cache. Toto prevedieme príkazom ping z počítača útočníka na obeť a bránu.
-nasledujúce výpisy:

```
Útočník ping Brána
PING 10.1.1.1 from 10.1.1.212 : 56(84}bytes of data.
64 bytes from 10.1.1.1 : icmp_seq=0 ttl=128 time=1.3 ms
Útočník$ ping Obeť
PING 10.1.1.12 from 10.1.1.212 : 56(84) bytes of data.
64 bytes from 10.1.1.12: icmp_seq=0 ttl=255 time=5.2 ms
```

Teraz môžeme pomocou programu arpredirect falšovať všetky ARP odpovede týkajúce sa brány, takže sieťová prevádzka od obeti určený pre bránu skončí útočníka. Program arpredirect je súčasťou nášho Live CD so systémom Linux.

```
Útočník$ arpredirect -t 10.1.1.12 10.1.1.1
```

V predchádzajúcom prípade sme preposielali iba dáta od obeti, pokiaľ bude chcieť preposielať všetky data určene bráne zadáme príkaz takto:

```
Útočník$ arpredirect 10.1.1.1
```

Keď sa pozrieme na ARP Cache u obeti, zistíme že sa zmenil.

```
Live root # arp
Address HWtype HWaddress Flags Mask Iface
10.1.1.212 Ether 00:0C:E6:C7:29:8A C eth0
10.1.1.1 Ether 00:0C:E6:C7:29:8A C eth0
Live root #
```

Zachytené dáta môžeme teraz analyzovať a priebežne preposielať na pôvodný cieľ. K tomu musíme mať na útočníkovi zapnutú podporu pre smerovanie, môžeme použiť smerovanie vstavane do jadra alebo špeciálny program. Smerovanie vstavane do jadra je v tomto prípade horšia voľba, pretože bude obeť na chybu v adresách upozorňovať pomocou ICMP redirectu. Lepšie je použiť špeciálny program napr. fragrouter, ktorý sa o presmerovanie paketov postará sám.

Na zmienenom príklade sme získali komunikáciu medzi dvoma PC. Pokiaľ ale chceme získať celu komunikáciu, robí sa to tak, že sa otvorí celá ARP Cache všetkých počítačov. To robi napríklad program Ettercap.

5.3.3 Obrana

Zmena cudzej ARP tabuľky sa dá uskutočniť veľmi jednoducho. Jednou zo základných obrán proti ARP presmerovaní je použiť statické ARP tabuľky. Medzi dôležitými systémami (napríklad firewallem a hraničnými smerovačmi) Ide použiť statické ARP tabuľky :

```
Oběť$ arp -s Brána 00:00:C5:74:EA:BC
Oběť$ arp -a
Brána (10.1.1.1) at 00:00:C5:74:EA:BC [ether] PERM on eth0
```

Vo výpise príkazu arp pribudla poznámka „PERM“, ktorá označuje ARP záznam. Zadávanie statických záznamov je ale nepraktické kvôli veľkému počtu počítačov. Preto je ďalší možnosť obrany proti ARP presmerovaní napríklad sledovanie IP adries a odpovedajúcich MAC adries v prípade zmeny môžeme včas zasiahnuť. O útoku sa len dozvieme, ide teda len o pasívne riešenie. Najbezpečnejším i keď pracnejším riešením je zadať MAC adresy priamo do prepínača. ARP mapy môžeme distribuovať bežnými spôsobmi, napríklad rdist, rsyns alebo /etc/ethers.

5.4 ÚTOK NA TCP SPOJENIE

Tento útok využíva nedostatočného zabezpečenia protokolu TCP. Útok spočíva v privlastnení si cudzieho, už vytvoreného pripojenia k inému sieťovému zariadeniu a následné možnosti komunikovať s týmito zariadeniami, ani by sa detekovalo, že už nekomunikuje s pôvodným klientom.

5.4.1 Sekvenčné čísla

Znalosť týchto čísel a dodržovanie číslovania je veľmi dôležité. Pokiaľ nejaké zariadenie obdrží TCP segment s spätným číslovaním sú tieto dáta neplatné. My pre naše prevedenie útoku potrebujeme tieto číslo poznať, čo znamená, že si ju musíme nejako zaopatriť.

Preto použijeme niektorú z technik pre odposluch dát na sieti. Tento útok sa môže tiež urobiť aj bez znalosti týchto čísel, pomocou tipovania, to je však pomerne náročné a nepoužívané. Ide tiež jednoducho zrušiť určité spojenie, keď poznáme sekvenčné čísla. Takému útoku sa hovorí desynchronizácia a spočíva v poslaní ľubovolných dát (najčastejšie binárnych núl) na klienta alebo server s tým, že sa vydávame za druhu stranu. To znamená, že obeť tieto dáta prijme, zmení informáciu o tom, aký bajt prijala, a informuje druhu stranu o prijatí týchto dát. Avšak tá o nami poslaných dátach nevie, a preto jej sekvenčné čísla nebudú sedieť. V tomto prípade vznikne tzv. ACK búrka a spojenie je desynchronizované, čo znamená, že už po ňom nebude možné prenášať žiadne dáta.

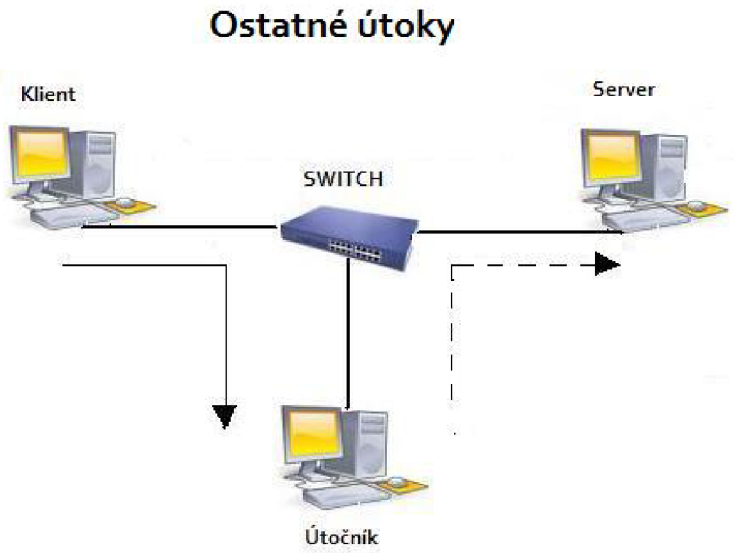
5.4.2 ACK Búrka

ACK búrka vznikne, pokiaľ sa sekvenčné čísla, ktoré si udržujú server a klient rozídu, čo sa normálne nestane, ale keď sa do hry pripojí útočník, môže táto situácia nastať. Keď potom server alebo klient pošlú nejaké dáta, tak ten druhý zistí, že sekvenčné čísla nesedia a pošlú TCP segment s nastaveným príkazom ACK a správne vyplnenými sekvenčnými číslami (pre každú stranu sú správne iné čísla).

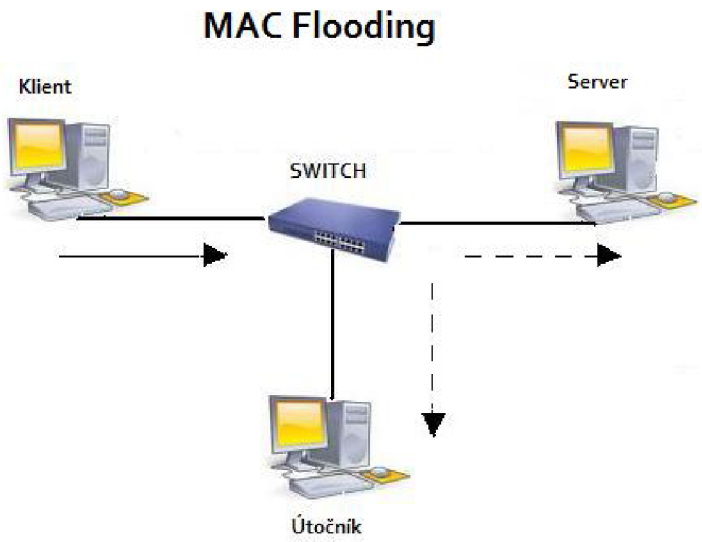
Avšak pre druhu stranu sú tieto čísla zlé, a tak urobí to isté(pošle TCP segment s príkazom ACK a správne nastavenými sekvenčnými číslami). Oba počítače sa ocitnú v nekonečnej smyčke pokiaľ sa jeden paket nestratí.

5.4.3 Prax

Je nutné poznať sekvenčné čísla, preto sa aplikuje napríklad útok ARP presmerovaním, tým sa docieli to, že uvidíme celkovú sieťovú prevádzku ľubovolného počítača umiestneného v lokálnej sieti. Pokiaľ sme na neprepínanej sieti (spojovacími sieťovými prvkami sú huby), potom nepotrebujem používať žiadny útok. Ale na neprepínané siete alebo pri použití útoku MAC Flooding budú vznikať ACK burky. Je to príčinou toho, že nemôžeme akýmkoľvek dátam zabrániť, aby bola doručená obr. 6.6



Obr. 5.6: Ostatné útoky



Obr. 5.7: MAC flooding

Akonáhle tečú všetky dáta cez nás, rozdelíme spojenie klienta so serverom na dve spojenia, a to klient - útočník a útočník – server.

Teraz obe dva spojenia budeme spravovať zvlášť. Útočník môže komunikovať so serverom plnohodnotne. Pokiaľ sa útočník rozhodne, že dá všetko do poriadku, musí synchronizovať sekvenčné čísla. To sa prevedie tak, že je na klienta poslaný potrebný počet znakov a zároveň je klient vyzvaný k zadaniu niekoľkých znakov. Toto je možne realizovať tak, že na klienta je poslaný napr. odkaz že došlo k výpadku prúdu a nech skúsi zadať niekoľko znakov.

Najjednoduchším spôsobom je toto spojenie rozdeliť na dve, rovnako ako v predchádzajúcom prípade a ihneď spojenie útočník – klient ukončiť. Server nič nepozná a klient si bude myslieť, že iba došlo k výpadku. Ďalšou možnosťou môže byť, že po rozdelení spojenia na dve budeme obhospodarovať obidve a to tým spôsobom, že dáta od klienta budeme preposielať serveru a vracať klientovi odpoveď od serveru a zároveň budeme komunikovať so serverom. Tento útok implementujeme napríklad programom Hunt.

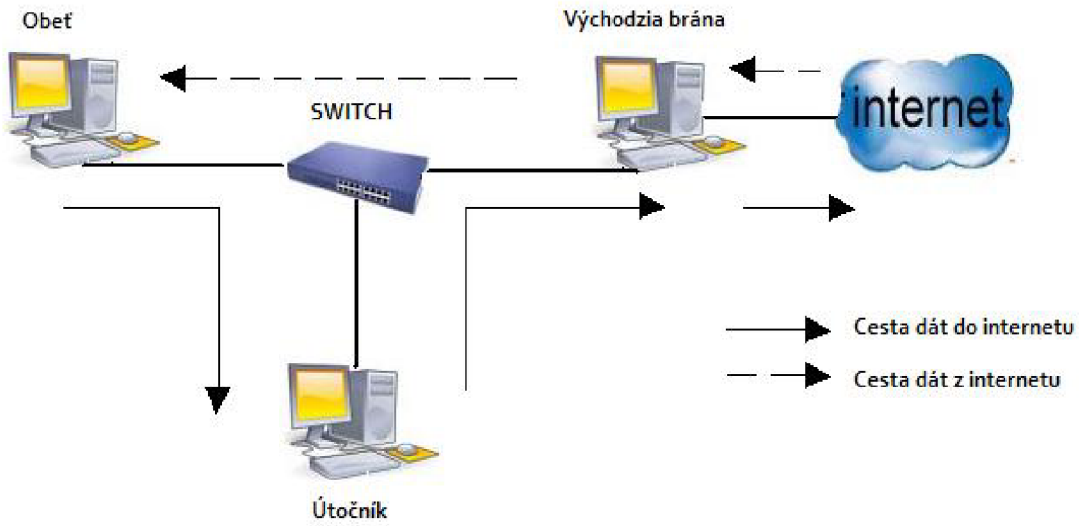
5.5 DHCP

DHCP (Dynamic Host Configuration Protocol) je aplikačný protokol pre automatické prideľovanie IP adries koncovým staniciam v sieti. Súčasne s IP adresou posiela server staniciam ďalšie nastavenia potrebné pre používanie siete ako je adresa najbližšieho smerovača (východzia brána), masku siete, adresy DNS serveru.

5.5.1 Teória

Útok využíva faktu, že na jednej sieti môže bežať viac DHPC serverov. Ďalší fakt, ktorý tomuto útoku pomáha je, že regulárne servery niesu príliš rýchle. Cieľom útočníka je sprevádzkovať a sieti nový DHPC server, a až si obeť spustí počítač tak jej podstrčiť nové údaje. V podstrčených údajoch môže byť falošná brána alebo DNS server.
Pokiaľ obeti podstrčíme falošnú bránu, bude komunikácia vyzerať, ako je znázornene na Obr.5.8 Ak podstrčíme falošný DNS server, môžeme vytvoriť útok zachycujúci oba smery toku dát.

Toho docielime tým, že na všetky dotazy bude odpovedať naša IP adresa a zo svojho počítača urobíme akoby proxy server.



Obr. 5.8: DHCP spoofing

Pokiaľ počítaču povieme pomocou útoku, že sme brána, bude cez nás posielať dáta do Internetu. Avšak dáta vracajúce sa z Internetu prídu na naozajstnú bránu a tá ju pošle cieľovému počítaču. Možnosť získať i dáta vracajúce sa z Internetu je ďalšou metódou útoku, alebo ide použiť NAT pre dáta, ktoré cez vás budú pretekať. Keď sa počítač pripojí do siete prvý krát, pošle a sieť paket DHCP (jedná sa o broadcast). Týmto paketom žiada, aby sa mi ozvali DHCP servery.

DHCP server mu odpovedá DHCP Offet, v ktorom mu ponúka parametry. Takto odpovie všetky DHCP servery. Avšak platí pravidlo najrýchlejšieho (záleží na implementácii DHCP klienta). Klient odpovedá najrýchlejšiemu serveru paketom DHCP Request, kde hovorí, že je všetko dohodnuté. Takto získal klient IP adresu. V tomto prípade by stačilo mať iba rýchlejší DHCP server a útok by sa podaril.

Pokiaľ už ale počítač niekedy bol v sieti pripojený, je postup iný. Počítač pošle iba DHCP Request serveru, od ktorého naposledy obdržal IP adresu. V pakete žiada o svoju poslednú IP adresu. Cieľový DHCP server mu žiadosť potvrdí paketom DHCP Ack (môže ju aj zamietnuť a poslať mu inú). Táto výmena sa odohrala bez toho, aby sme ju na prepínanie sieti počuli.

Dôležitým je tiež parameter „lease time“. Tento parameter určuje server a hovorí klientovi, ako dlho má danú IP adresu priradenú. Klient si musí vždy pred uplynutím tejto doby predĺžiť platnosť priradenia. Pokiaľ tak neurobí, server si označí danú IP adresu ako volnú a môže ju ponúknuť niekomu inému. Pokiaľ je už počítač v sieti pripojený robí útok tak, že vyčerpajú všetky IP adresy, ktoré DHCP server priraďuje. Akonáhle nemá DHCP server volné IP adresy pre priradenie prestane odpovedať na pakety DHCP Discover (ak nemá čo ponúknuť). Teraz musíme počkať nejaký čas, nech uplynie doba priradenia IP adries (lease time) obsadených počítačov a zaberieme ich tiež.

Teraz, keď sa počítač spustí a bude žiadať o svoju starú IP adresu, nedostane žiadnu odpoveď alebo dostane zápornú odpoveď. V tomto prípade väčšina klientov pošle do siete DHCP Discover a chovajú sa akoby v sieti ešte nikdy neboli.

Nastane opäť miesto pre náš falošný DHCP server, ktorý môže ponúknuť napríklad zabrané IP adresy, čiže sa nemusíme obávať, že by nastal nejaký konflikt. Keď už budeme mať nachystaných klientov, ktoré sme potrebovali, môžeme útok na regulárny DHCP server ukončiť

5.5.2 Prax

Útok začneme vysadením volných adries z regulárneho DHCP servera. Pre tento účel nám poslúži nástroj „dhcpx“ z balíku IRPAS. Spustíme ho

./dhcpx -vv -i rozhraní -A –D IP_adresa_regulárneho_DHCP_server.

Pomocou programu Ettercap môžeme spustiť DHCP spoofing. Parametry nastavíme podľa situácie a nastavenia miestnej siete (ako gateway sa automatický zvoli vaša IP adresa).

5.5.3 Obrana

Najúčinnejšou obranou je nepoužívať DHCP server a používať staticku konfiguráciu, čo je ale nepraktické a zdĺhavé. Preto tomuto útoku môžeme zabrániť tak, žš nastavíme hodnotu „lease time“ napríklad na týždeň alebo aj na mesiac či viac.

Lease time určuje ako dlho bude IP adresa priradená obetí a nedostane ju nikto iný. Pokiaľ je počítač zapnutý tak si toto priradenie predlžuje (obnovuje). Jediná možnosť ako by útočník mohol napadnúť takýto počítač pomocou svojho DHCP serveru je ta, že by počítač nebol prítomný v sieti alebo bol vypnutý po dobu „lease time“. Sieť s takýmto nastavením DHCP serveru by sa javila ako sieť s statickými IP adresami. Preto sa toto nastavenie nehodí tam, kde niesu počítače nastálo a často sa striedajú, tu by mohlo dochádzať k nedostatku IP adries. Administrátor by mal nadefinovať porty, za ktorými sa nachádza DHCP server. DCHP komunikácia je potom povolená len na týchto portoch. Prepínač si naviac môže z informácií získaných z DHCP paketu vybudovať tabuľku, v nej je uvedená pre každú pripojenú väzbu medzi MAC adresou, IP adresou, dobou pridelenia IP adresy, portom prepínača a VALN sieti. Túto tabuľku ide použiť pri ochrane proti ďalším útokom, ako napríklad proti pokusom nedovolene meniť zdrojové IP adresy paketov.

5.6 UDP flood

Útok je založený na zasielaní UDP paketov s falošnou adresou odosielateľa. Keďže UDP je protokol bez nadväzovania spojenia, útočník zasiela UDP pakety na náhodné porty cieľového počítača. Ak počítač prijme UDP paket adresovaný na port, ktorý nevyužíva nijaká aplikácia, zašle odosielateľovi ICMP paket s hlásením o chybe. Ak útočník použije

dostatočne veľa UDP paketov, na sieti vznikne veľká (dvojnásobná) prevádzka a počítač zamrzne.

5.6.1 Využitie Vašich vlastných prostriedkov na DoS

Zaujímavý útok je opísaný v knihe [3]. Útočník používa falošné UDP pakety na prepojenie dvoch počítačov pomocou služieb "chargen" (generovanie náhodných znakov) a "echo" (vypíše to, čo dostane na vstup) - zašle na prvý cieľový počítač UDP paket pre službu "chargen" so zdrojovou adresou druhého cieľového počítača a portu pre službu "echo". Druhý cieľový počítač bude pomocou služby "chargen" generovať pakety a zasielať ich službe "echo" na prvý cieľový počítač. Tieto dve služby zahltia celé dostupné pásmo medzi počítačmi. Útočník vystupuje "iba" v úlohe iniciátora. Ochrana proti tomuto konkrétnemu útoku je vypnutie (v skutočnosti málo využívaných) služieb "echo" a "chargen", napríklad na úrovni "`inetd`" démona, ktorý ich zväčša obsluhuje. Samozrejme, pomôže aj filtrovanie týchto služieb na firewalli.

5.6.2 Zahltenie linky (flood)

Útočník začne posielať veľké množstvo paketov (napr. ICMP ECHO) na cieľový server. Pakety zvyknú mať falošnú IP adresu odosielateľa a často sú zasielané z niekoľkých serverov súčasne. Cieľový server začne zasielať odpovede a dôjde k zahlteniu linky.

5.6.3 Útoky "Ping of death", "Tear drop" a "Bonk"

Tieto útoky využívajú ICMP pakety. "Ping of Death" spočíva v zasielaní príliš veľkých ICMP ECHO paketov, ktoré preplnia vyhradenú pamäť pre paket na cieľovom počítači. Útoky "Tear drop" a "Bonk" sú založené na prekrývajúcich sa fragmentoch ICMP paketov (zlý offset) a slúžia najmä na odstavenie počítačov s operačným systémom Windows 98. Protiopatrením je zavedenie povinnej defragmentácie na routeroch alebo vo Vašom kerneli (voľba "IP always defragment"), ak funguje Váš Linuxový server ako brána do Internetu. Ďalšou možnosťou je blokovanie fragmentov ICMP paketov na firewalli.

5.6.4 Ochrana proti sieťovému DoS útoku

1. **vypnite služby, ktoré nepoužívate**, neposkytnete tak potenciálnym útočníkom potenciálne ciele útoku
2. **zapnite v kerneli overovanie zdrojovej cesty** pomocou príkazu

```
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 1 > $f; done
```

   v štartovacích skriptoch, alebo si naštudujte spôsob, ako to robí Vaša distribúcia. Táto možnosť v kerneli zabezpečí, že cez sieťový interface neprejdú pakety s adresou odosielateľa, ktorá na tejto strane siete nemá čo hľadať - napr. neprepustí pakety s privátnymi adresami prichádzajúce z vonkajšej siete

3. **firewall**:

   a.) paketový filter (`ipchains`, `iptables`):

   umožňuje predchádzať mnohým DoS útokom tým, že neprepustí pakety definované na základe adries a portov. Paketový filter zahadzuje pakety ešte skôr, ako sa k nim dostane obslužný démon, takže odpadajú problémy so zvýšeným logovaním alebo zaťažením procesora. Ja zvyknem filtrovať všetky pakety, ktoré smerujú na neobsadené porty alebo filtrujem pripojenie na služby, ktoré majú byť prístupné iba z vnútornej siete

   b.) limit na počet spojení (`iptables`):

   umožňuje nastaviť pravidlá s limitmi pre počet spojení za jednotku času. Pomocou týchto pravidiel sa vyhnete mnohým DoS útokom a to aj pre služby, ktoré musia byť prístupné aj z verejnej siete (WWW, mail, DNS, ...)

4. **pre útoky založené na ICMP** zapnite defragmentáciu paketov vo Vašom kerneli (IP always defragment) - pozor, iba v prípade, že je Váš Linuxový server jedinou cestou pre pakety.

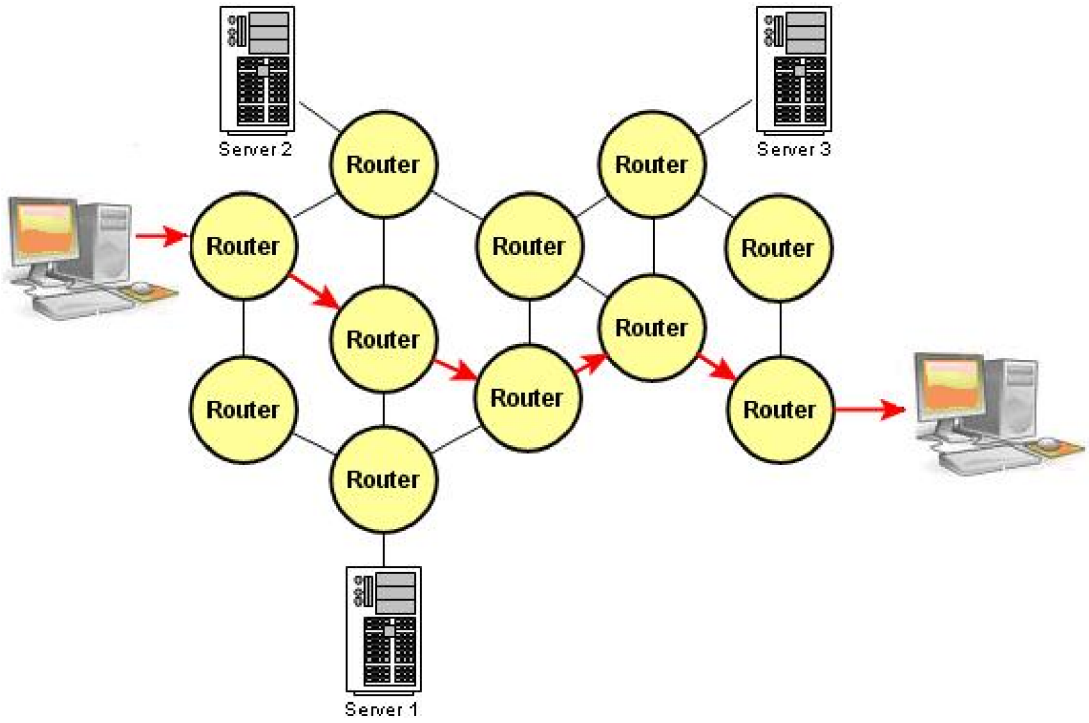
5. **zapnite možnosť TCP_SYNCOOKIES** vo Vašom kerneli (2.2.x, 2.4.x) a potom (a pri každom štarte servera) vykonajte:

```
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

Podľa dokumentácie v kerneli budete lepšie chránení proti SYN floodu - resp. server bude fungovať aj počas syn floodu.

5.5 Denial of service: zaplnenie disku

Okrem sieťových DoS [10] útokov existujú aj ich varianty, ktoré používajú iné prostriedky, ktorých množstvo je obmedzené, najmä diskovú kapacitu. Útočník môže začať generovať také pokusy o pripojenie alebo vyvinie inú aktivitu, ktorá má za následok zvýšené logovanie. Ak logy zaplnia všetok dostupný priestor, môže dôjsť k zastaveniu niektorých služieb, napr. mail servera (ak server nemá kam prijímať poštu). Iným variantom tohto útoku je zasielanie veľkého množstva mailov, ktoré zaplnia disk. Následkom zaplnenia disku môžu prestať fungovať aj iné služby, ktoré potrebujú na disk zapisovať. Nemusíme však chodiť až tak ďaleko - server dokáže znefunkčniť aj bežný používateľ. Pri útoku sa využíva fakt, že používateľ nemá obmedzené miesto na disku a môže jeho zaplnením obmedziť činnosť ostatných používateľov alebo - čo je ešte horšie - činnosť celého systému. To sa môže stať v prípade, že sa používateľské údaje (domáce adresáre) alebo dočasné súbory (do ktorých má právo zapisovať každý) nachádzajú na tom istom súborovom systéme, ako systémové údaje, ktoré server počas svojej prevádzky mení (logy, databázy, dočasné súbory, ...). Ak server nemá kam zapísať údaje do systémového logu, neexistuje možnosť monitorovania servera. Ak niektorá služba netestuje možnosť, že nemá kam zapísať údaje, môžu nastať nepredvídateľné okolnosti, ktoré s najväčšou pravdepodobnosťou vedú k strate dát a k zastaveniu prevádzky služby.



Obr. 5.9: Denial of service

5.5.1 Ochrana proti zaplneniu disku

1. Používanie oddelených súborových systémov [13] pre:
   - systémové údaje,
   - používateľské údaje,
   - dočasné údaje,
2. používanie súborových kvót.

5.5.2 Denial of Service: zneužitie prostriedkov servera používateľmi

Na záver si spomenieme ďalší spôsob zastavenia služieb, ktorým je zneužitie prostriedkov servera lokálnymi používateľmi. Do tejto kategórie patrí zaplnenie pamäte, enormné zaťaženie procesora, zaplnenie tabuľky procesov a podobne. Štandardne systém neobmedzuje počet používateľských procesov a ich veľkosť v pamäti. Veľkosť pamäte je však obmedzená a ak používateľský program zaplní všetku dostupnú pamäť, server "umrie". To isté sa stane v prípade, že používateľ zaplní tabuľku procesov (počet súčasne bežiacich

procesov je obmedzený). Ak používateľ bude vykonávať zložité operácie, jeho procesy zaťažia server natoľko, že môže prestať odpovedať na požiadavky.

5.5.3 Ochrana proti zneužitiu prostriedkov servera

Pred chvíľou sme spomenuli súborové kvóty [11]. Bolo by dobré, keby niečo podobné existovalo aj pre procesy, pamäť atď. No, a niečo podobné, prirodzene, existuje.

PAM (Pluggable Authentication Modules) [12] je systém, ktorý oddeľuje programy od autentifikácie/prihlásenia pomocou jednotného rozhrania a umožňuje tvorbu modulov pre rôzne metódy prihlásenia. Okrem samotného prihlásenia sa tieto moduly môžu starať aj o iné záležitosti.

Heslo, ktoré nie je jednoduché uhádnuť je základ bezpečnosti. Ako seba a ostatných donútiť zvoliť takéto heslo? V tom nám pomôže systém PAM a modul pam-passwdqc
Tento sa nachádza v balíčku libpam-passwdqc (Ubuntu, Debian). Po inštalácii je potrebné nastaviť aby sa tento modul používal.
Upravíme konfiguračný súbor `/etc/pam.d/common-password`
(tento by mal byt spoločný pre všetky programy, ktoré umožňujú meniť užívateľské heslo):

```
# pôvodné nastavenie
#password required pam_unix.so nullok obscure min=4 max=8 md5

# nové nastavenie
password   required   pam_passwdqc.so   min=disabled,16,12,8,6
max=256
password sufficient pam_unix.so use_authtok md5
password required pam_deny.so
```

Parameter `min` pre minimálne požiadavky na heslo sme nastavili nasledujúcim spôsobom (typy znakov sú veľké písmena, malé písmená, číslice a ostatné znaky):
disabled zakázali sme heslá tvorené len jedným typom znakov (akejkoľvek dĺžky) 16 minimálna dĺžka hesla pozostávajúceho z dvoch typov znakov 12 je nastavenie dĺžky pre "passphrase" (modul náhodne vygeneruje vyhovujúcu "passphrase") 8 dĺžka hesla

pozostávajúceho z troch typov znakov 6 dĺžka hesla pozostávajúceho zo štyroch typov znakov.
Podrobnú dokumentáciu nájdeme v adresári "/usr/share/doc/libpam-passwdqc"

Ak Váš server podporuje PAM, pomocou modulu "pam_limits.so" môžete zabrániť mnohým lokálnym DoS útokom. Stačí, ak v konfiguračnom súbore "/etc/pam.d/ssh" (predpokladám, že sa na server prihlasujete len pomocou SSH [13] a to je aj jediný spôsob, ktorý Vám odporúčam, okrem administrátorskej práce s konzolou) doplníte tieto riadky:

```
session     required      /lib/security/pam_limits.so
session     required      /lib/security/pam_unix.so
```

Potom upravte konfiguračný súbor "/etc/security/limits.conf". Jednotlivé riadky majú tvar:

kto typ čo hodnota

- **kto**: určuje používateľa alebo skupiny používateľov (začínajú znakom "@"), pre ktorých platí dané obmedzenie (môžete použiť znak "*" ako zástupný znak pre "všetkých" používateľov. Nerobte to, ak neviete, čo robíte, pretože obmedzíte aj procesy bežiace pod rootom.)

- **typ**: určuje typ obmedzenia ("**hard**": tvrdý limit, nemožno ho obísť, "**soft**": limit sa dá obísť až po hodnotu hard - presne to otestované nemám, ale pre naše účely budeme používať iba hodnotu "hard")

- **čo**: určuje, čo budeme obmedzovať. Tu sú najdôležitejšie hodnoty:
  - **core**: veľkosť súboru "core" (po páde procesu),
  - **fsize**: maximálna veľkosť súboru (KB),
  - **rss**: maximálna veľkosť procesu v pamäti (KB),

- **nproc**: Maximálny počet procesov (pamätajte, že používateľ potrebuje minimálne 2 procesy, aby sa vôbec prihlásil pomocou SSH: "sshd" a shellovský proces.) .

- **hodnota**: Určuje číselnú hodnotu pre dané obmedzenie.

Príklad: obmedzíme skupinu "students" nasledovne: nulová veľkosť súboru "`core`" (pri páde niektorej aplikácie by sa zbytočne zapĺňal diskový priestor), proces môže zaberať v pamäti maximálne 16 MB (16384 KB) a používateľ môže spustiť maximálne 10 procesov súčasne.

```
@students       hard    core    0
@students       hard    rss     16384
@students       hard    nproc   10
```

Podobne môžete vytvoriť obmedzenia aj pre ďalšie skupiny používateľov.

6. Záver

Problematika sieťovej bezpečnosti nie je ľahko zmapovateľná, pretože neustále rozvíjajúce sa technológie posúvajú tieto témy každým dňom na inú, vyššiu úroveň.

Hlavným cieľom bolo podrobne popísať problematiku bezpečnosti na operačný systém Linux. Získať dostupné informácie o najviac používaných útokoch a venovať sa problematike bezpečnosti. Zle nakonfigurovaný Linux je rovnako nebezpečný, ako zle nakonfigurovaný iný UNIX-like systém. Nazbierané teoretické informácie som využil v praktických ukážkach týchto útokov. Nasimuloval som útoky SYN flood, Buffer overflow, ARP presmerovanie, DHCP, ACK. Zhodnotil som mieru nebezpečnosti a to mi pomohlo nájsť riešenia, aké kroky podniknúť, aby sa tieto útoky sa neopakovali.

Myslím si, že téma „bezpečnosť“ najmä v dnešnej dobe sa bude vážne riešiť, nakoľko vývoj informačných technológií rapídne rastie a to ma potom za následok vymýšľať nové nebezpečné útoky nielen na systém Linux, ktorý bol v roku 2007 naviac napadnutý serverový systém, ale aj na iné operačné systémy.

Použitá literatúra

[1] WHEELER, David A. . Secure Programming for Linux and Unix HOWTO [online] 1999-2003 [cit. 2007-12-09]. Dostupný z WWW: < http://www.dwheeler.com/secure-programs/secure-programs-OWTO/index.html>

[2] ZWICKY, Elizabeth D., COOPER, Simon , CHAPMAN, D. Brent . Building Internet Firewalls. [s.l.] : [s.n.], 2000. 894 s.

[3] CHESWICK, William R., BELLOVIN, Steven M. , RUBIN, Aviel D. Firewalls and Internet Security. [s.l.] : [s.n.], 2002. 464 s. ISBN 0-201-63466-X.

[4] ERICKSON, Jon. Hacking: The Art of Exploitation . [s.l.] : Zoner Press, 2005. 263 s.

[5] WIMER, Scott . Preventive Security [online]. 2002 [cit. 2007-12-09]. Dostupný z WWW: <http://freshmeat.net/articles/view/434>.

[6] CHABREČEK , Miroslav . Bezpečnosť operačného systému GNU/Linux. [s.l.], 2007. 70 s. UNIVERZITA KOMENSKÉHO V BRATISLAVE. Diplomová práce.

[7] NORIS, Ivan. Bezpečnosť servera v sieti [online]. 2002-2007 [cit. 2007-12-09]. Dostupný z WWW: <http://deja-vix.sk/sysadmin/security.html>.

[8] Jak odhalit útok [online]. 2004 [cit. 2007-09-26]. Dostupný z WWW: <http://www.owebu.cz/bezpecnost/vypis.php?clanek=345>.

[9] MACKO, Rastislav. Bezpečnost v GNU/Linuxu, prednáška [online]. 2004 [cit.2007-11-07].Dostupný z WWW: <http://avc.siliconhill.cz/archiv/index.php?id=1041&rid=17&offset=0&select=InstallFest>.

[10] HALLER, Martin . Denial of Service (DoS) útoky: úvod [online]. 2006 [cit. 2007-09-30]. Dostupný z WWW: <http://www.lupa.cz/clanky/denial-of-service-dos-utoky-uvod />.

[11] NORIS, Ivan. Rozdelenie disku [online]. 2005 [cit. 2007-12-09]. Dostupný z WWW: <http://mirror.7crows.net/psa/fs.html#quotas>.

[12] NEMETH, E., SNYDER, G., HEIN, T. R. Linux : Kompletní příručka administrátora. [s.l.] : Computer Press, 2004. 880 s.

[13] HATCH, Brian, LEE, James , KURTZ, George. Linux Hackerské Útoky. [s.l.] : SoftPress, 2002. 576 s. Mimo edice. ISBN 80-86497-17-8.

[14] Linux Box Security [online]. 2007 [cit. 2007-12-09]. Dostupný z WWW: <http://blackhole.sk/topiclinux-box-security-part-1-introduction>.

[15] KRČMÁŘ , Petr . Linux – tipy a triky pro bezpečnost. [s.l.] : Grada, 2004. 208 s. ISBN 8024708124.

[16] NORIS , Ivan. *Sieťové služby* [online]. 2002-2007 [cit. 2007-12-09]. Dostupný z WWW: <http://mirror.7crows.net/psa/services.html#inetdvsstandalone>.

[17] *LinuxOS* [online]. 2005 [cit. 2008-05-22]. Dostupný z WWW: <http://www.linuxos.sk/clanok_pdf/285/index.html>.

[18] HERBORTH, Chris . *Unix a Linux - Názorný průvodce*. [s.l.] : [s.n.], 2006. 288 s. ISBN 80-251-0978-X.

[19] TOXEN, Bob. *Bezpečnost v Linuxu - Prevence a odvracení napadení systému*. [s.l.] : [s.n.], 2003. 876 s. ISBN 80-7226-716-7.

Zoznam skratiek

ACLs	Access Control Lists
BSD	Berkeley Software Distribution
DAC	Discretionary Access Control
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name systém
DOS	Disk Operating systém
DoS	Denial of Service
EIP	Extended Instruction Pointer
ICMP	Internet Control Message Protocol
LIDS	Linux Instrusion Detection System
MAC	Mandatory Access Control
NSA	National Security Agency
PAM	Pluggable Authentication Modules
PGP	Pretty Good Privacy
SFP	Saved Frame Pointer
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol