Rootkity

* rootkit je balíček programů, který dovoluje útočníkovi skrýt své aktivity v systému, přistupovat k němu podle potřeby apod.
 - ve většině případů útočník po napadení systému nainstaluje rootkit
 - obsahuje různé programy používající již uvedené koncepce (trojský kůň, zadní vrátka apod.)
 - instaluje do adresáře, který pravděpodobně nebudete prohledávat, např. "...", ".. ", /usr/lib/libsof apod.

* rootkity obvykle obsahují:
 - program /bin/login se zadními vrátky, který útočníkovi dovolí pozdější přihlášení jako root
 - program pro odposlech paketů a program pro zamaskování odposlechu paketů
 - programy pro zamaskování souborů a procesů útočníka

* program pro odposlech paketů a program pro zamaskování odposlechu paketů
 - při odposlechu paketů je rozhraní přepnuto do promiskuitního režimu
 - můžeme detekovat pomocí /sbin/ifconfig (vypíše příznak PROMISC)
 - proto v rootkitu falešná verze programu ifconfig, která příznak PROMISC nezobrazuje

* v rootkitech pro UNIXové systémy jsou nové verze programů ps, ls, netstat
 - ps - vypisuje procesy, verze z rootkitu nevypisuje procesy útočníka
 - ls - vypisuje obsah adresáře, verze z rootkitu nezobrazuje soubory rootkitu
 - netstat - informace o síťovém rozhraní, skrývá aktivity útočníka, tj. jeho privilegované procesy poslouchající na portu apod.

* obrana:
 - kontrola integrity kritických programů (login, ps, ls, du, netstat, ...) + knihoven (libc.so.6)
  . např. tripwire - generuje jednosměrný hash spustitelných souborů
  . administrátor může uchovat na read-only médiu (CD ROM, R/O disketa) a periodicky kontrolovat
  . pokud je detekován rootkit, nejbezpečnější je nainstalovat OS (+ případné kritické aplikace) znovu z čistého zdroje

* poslední vývoj - rootkity na úrovni jádra
 - využívají mechanismus za běhu zaveditelných modulů (Linux, Solaris) => není zapotřebí restart systému
 - nejoblíbenější vlastnosti:
  . přemapování požadavků na spuštění programu - místo požadovaného programu spustí program z rootkitu => nepomůže kontrola integrity původního programu
  . skrývání souborů - útočník si může zvolit, který soubor v systému nebude pro uživatele jádra existovat
  . skrývání procesů - podobně jako výše

* obrana proti rootkitům na úrovni jádra
 - částečná obrana: na kritických strojích zakázat moduly zaveditelné za běhu

Všeobecná obrana

* nejdůležitější z popsaných strategií
 - neprovozovat zbytečné systémy a služby
 - sledovat a instalovat opravy SW od výrobců

* kromě dříve popsaných konkrétních strategií obrany musí být organizace schopna detekovat napadení svých systémů a reagovat na něj

* často se používají systémy pro detekci napadení (intrusion detection systems, IDS)
 - obsahují databázi příznaků napadení, porovnávají s ní provoz na síti
 - umožňuje detekovat útok v počátečním stádiu

* ještě důležitější - definovat a popsat reakci organizace na útok

```
 - pokud se reaguje až ve chvíli krize, budou zodpovědné osoby dělat chyby
 - proto by měl existovat dokument popisující roli jednotlivých osob a
   sekvenci akcí (sekvence příkazů, případná komunikace s policií apod.)
 - pokud je bezpečnost kritická, doporučuje se konat pravidelná cvičení

* penetrační testování = určení schopnosti subjektu odolávat útokům
 - tester používá stejné triky a techniky jako reálný útočník
 - dovoluje najít slabá místa dříve, než je najde útočník
* obvyklé jsou následující typy testů:
 - testy fyzické infrastruktury subjektu
  . je možné se dostat nekontrolovaně do budovy? (test: pokus dostat se do
    budovy v době, kdy zaměstnanci přicházejí do práce)
  . jsou citlivé části budovy uzamčeny nebo mají jinou fyzickou bariéru?
  . je možné budovu nekontrolovaně opustit? (test: pokus pronést laptop apod.)
 - testy operačních aspektů organizace
  . jaká jsou pravidla pro změnu hesla apod., lze obsluhu přesvědčit bez
    důkazu identity, aby vytvořila účet, změnila heslo nebo přístupová práva?
  . existují pravidla jak zacházet s vyřazovanými datovými nosiči nebo s
    disky v počítačích předávaných do opravy (např. rušení dat) a jsou
    dodržována? jaké citlivé informace lze na nich nalézt?
 - elektronické testy - použití už zmíněných nástrojů
* množství lidí kteří o testu vědí by mělo být minimalizováno (důležité je
  aby test schválilo vedení!)


Viry a červi
============

* viry = programy, které se reprodukují připojením svého kódu k jinému
  programu; virus je aktivován spuštěním nakaženého programu
* červi = programy, které se samy replikují a šíří (tj. nepotřebují, aby
  uživatel program spustil)

* oba typy mohou vykonávat i jinou činnost než se jen šířit:
 - neškodné činnosti (vypsat zprávu apod.)
 - poškodit uživatele
  . poškodit nebo zrušit soubory, zaslat soubory někam, zašifrovat apod.
  . způsobit nepoužitelnost počítače (DoS, např. v UNIXu: while (1) fork();)
  . permanentně poškodit HW (např. flash ROM)
 - v mnoha případech nedělá nic pozorovatelného do určitého data

* autoři virů i červů většinou středoškoláci nebo vysokoškoláci, kteří jejich
  napsání považují za ”zajímavou věc, na které se něco naučí” (většinou si
  neuvědomují (nebo je nezajímá) jaké tím mohou v sumě napáchat škody)

Viry
----

[ bylo uvedeno v referátu ]

MS plánuje ”code signing”:

- kód s digitálním podpisem, vytvořit umí pouze ”certifikované entity”,
  ověřit umí kdokoli
- sdružení jména autora (firmy) s kódem, není možná změna kódu aniž by
  ”podpis” přestal platit
- volba OS ”accept only signed code”

Pro malé vývojáře a individua je vydávání digitálních certifikátů obtížné a
drahé (nutno ověřit skutečnou identitu) => je dražší než pro velké firmy =>
možné vyloučení konkurence ze hry?

* ve víceuživatelských OS jsou adresáře obsahující spustitelné programy
  nepřístupné pro zápis => viry obvykle nejsou problém

Červi
-----

* červ = program, který se šíří prostřednictvím počítačové sítě nezávisle na
```

činnosti uživatele
- obvykle využívá slabá místa v běžném SW, z napadeného počítače se šíří dále
- některé dnešní viry používají kombinaci technik virů a červů

Internet Worm
....

* asi nejznámější bezpečnostní incident - Internet Worm (1988)

* Robert Morris našel 2 chyby v BSD UNIXu, které umožňovaly neautorizovaný přístup na stroje v Internetu
 - napsal samoreplikující se program, který chyby využíval
 - na programu pracoval několik měsíců, uvolnil 2. listopadu 1988
 - během několika hodin od uvolnění vyřadilo z provozu většinu strojů Sun a VAX připojených na Internet
 - Morrisova motivace není známa, pravděpodobně vtip který se vymknul z rukou autora

* červ se skládal ze 2 programů - zavaděče a vlastního červa
 - zavaděč - program l1.c (99 řádků v C), na napadeném systému se přeložil a spustil, spojil se se strojem odkud byl zaslán, přečetl a spustil vlastního červa
 - červ prohlédl směrovací tabulky, našel a napadl další stroj
 - pro napadení dalších strojů 3 metody:
  . zkusit rsh na jiný stroj
  . daemon fingerd používal pro čtení požadavku fci gets(), poslat speciálně vytvořený 536 bytový řetězec který přepsal zásobník - návrat do procedury v řetězci, která spustila /bin/sh
  . chyba v poštovním programu sendmail, která dovolovala poslat a spustit kopii zavaděče
 - v napadeném systému se pokusil rozluštit hesla (v /etc/passwd) - možnost přihlásit se všude, kde měl uživatel stejné heslo
 - pokud na daném stroji červ již běžel, nová kopie skončila v 6 případech ze 7, ale to způsobilo příliš mnoho běžících červů => DoS

* Morris byl odsouzen, 10 000$ pokuty, 3 letá podmínka + 400 hodin veřejné služby
 - pozitivní efekt - vytvoření CERT (Computer Emergency Response Team) - sbírá informace o problémech OS a o způsobu opravy, v případě potřeby zveřejňuje

Další vývoj
....

* současní červi např. Ramen, L10n, Cheese, Code Red, Nimda

* rychle se šířící červi
 - poslední červi byli zatím poměrně neefektivní, protože prostor IP adres prohledávaly náhodně
 - v reakci na to se objevily články popisující efektivnější techniky - předprohledání Internetu na napadnutelné systémy atd.
 - umožnilo by do 1 hod napadnout 99% napadnutelných systémů
 - zatím naštěstí nebylo realizováno
* víceplatformoví červi
 - většina červů zatím napadá jeden typ systému, především Windows nebo Linux
 - lze očekávat, že budoucí červi budou napadat více typů OS
* morfující a maskující se červi
 - dosavadní viry bylo možno relativně snadno detekovat
 - lze očekávat polymorfní červy, jejichž kód se bude měnit s každým napadením systému
* destruktivní červi
 - dosavadní červi nenesli destruktivní kód, což se může změnit

* obrana proti červům
 - včasná instalace bezpečnostních oprav OS a dalších programů
 - pokud je to možné, zásobník by neměl být spustitelný (alespoň pro aplikace, které mají fci serveru)
 - blokovat nepotřebná výstupní spojení (aby se červ nemohl dále šířit)
 - mít vypracován plán odpovědi na incident

Důvěryhodné systémy
==========

* většina posledních přednášek popisovala, že téměř všechny moderní OS nejsou bezpečné (”they leak like a sieve”)
* v souvislosti s tím bychom se mohli ptát:

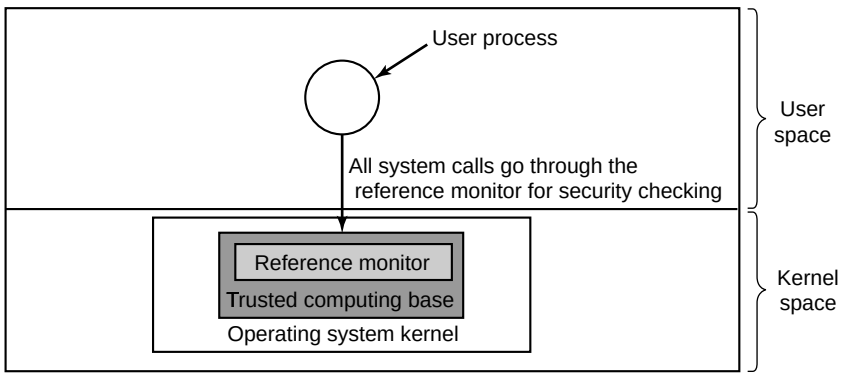
1. Je možné vytvořit bezpečný počítačový systém?
2. Pokud ano, proč se to nedělá?

* odpověď na (1) je v zásadě ano, základní zásady jsou dlouho známy, např. MULTICS je příklad OS vytvořeného s cílem bezpečného systému, celkem se povedlo
 - mechanismus musí být nejnižších vrstvách OS, musí být dostatečně jednoduchý a nesmí být obcházen
* proč (2), tj. proč nejsou bezpečné má v zásadě dvě příčiny:
 - současné systémy nejsou bezpečné, ale většině uživatelů ”stačí”
 - bezpečný OS musí být jednoduchý, ale uživatelé požadují funkčnost => více kódu => více chyb => více bezpečnostních chyb
  . například první e-mailové systémy posílaly pouze textové zprávy, byly zcela bezpečné
  . později jiné typy dokumentů => např. dokumenty ve Wordu, mohou obsahovat makra = na počítači spouštíte cizí kód
  . jiný příklad: dříve se web skládal z pasivních WWW stránek, dnes pro prohlížení stránek často nutné spouštět applety

TCB
...

* pro některé organizace je bezpečnost primární, podíváme se na některé modely
 - místo ”bezpečné systémy” se v oblasti bezpečnosti OS mluví o ”důvěryhodných systémech”
 - základem každého důvěryhodného systému je HW a SW potřebný nebo ovlivňující vynucování bezpečnostních pravidel, který se nazývá TCB (Trusted Computing Base)
 - TCB typicky zahrnuje většinu HW (kromě I/O zařízení neovlivňujících bezpečnost), část jádra (vytváření procesů, přepínání mezi procesy, správa mapy paměti, část správy I/O), většinu programů s přístupovými právy administrátora
 - v důvěryhodných systémech je často TCB oddělena od zbytku OS z důvodu minimalizace její velikosti a ověření správnosti

* důležitou částí TCB je monitor odkazů (reference monitor)
 - rozhoduje o všech požadavcích týkajících se bezpečnosti (např. otevření souboru) zda mohou být vykonány
 - dovoluje, aby se všechna rozhodnutí děla na jednom místě bez možnosti je obejít
 - většina OS není navržena tímto způsobem = jeden z důvodů proč nejsou bezpečné



Matice ochrany

* v OS objekty, které je třeba chránit před neoprávněným přístupem (soubory, procesy, semafory, segmenty paměti, I/O zařízení)
 - každý objekt má jedinečné jméno kterým je odkazován a konečnou množinu operací, které je nad ním možno provést
 - procesy v každém časovém okamžiku běží v některé doméně
  . doména obvykle odpovídá jednomu uživateli
  . doména určuje podmnožinu operací, které je možné vykonat nad objektem
  . povolené podmnožině operací říkáme ”přístupová práva”
 - například v systému UNIX jsou přístupová práva k souborům podmnožinou { Read, Write, eXecute }, doména je určena UID a GID procesu

* popis mechanismů ochrany se zavádí pojem matice ochrany
 - řádky = domény
 - sloupce = objekty
 - aby bylo možné přepínat mezi doménami (např. měnit UID a GID), jsou domény v matici ochrany uvedeny také

	Object										
	File1	File2	File3	File4	File5	File6	Printer1	Plotter2	Domain1	Domain2	Domain3
main 1	Read	Read Write								Enter	
2			Read	Read Write Execute	Read Write		Write				
3						Read Write Execute	Write	Write			

 - ve skutečnosti není implementováno jako matice, nejčastěji implementováno jako ACL:
 . s každým objektem OS sdruženo ACL, v něm seznam dvojic (doména, přístupová práva)
 . doména např. UID a GID, přístupová práva - bitové pole, 0=odepřeno, 1=povoleno
 . příklad ACL v UNIXu:

```
objekt        ACL
------------------------------
prednaska.txt luki, staff: RW-
              *, staff:    R--
              *, students: R--

/etc/passwd   root, root:  RW-
              *, *:        R--
```

Formální modely důvěryhodných systémů

* matice ochrany nejsou statické, obvykle se mění při vytváření nových objektů
 - nad maticí ochrany 6 primitivních operací nad maticí ochrany, kterými lze modelovat jakýkoli systém: create object, delete object, create domain, delete domain, insert right, remove right (Harrison et. al. 1976)
 - tato primitiva mohou být součástí služeb OS (”příkazů”), které mohou vyvolávat uživatelské programy
  . např. vytvoření nového souboru: vytvoření objektu souboru, do matice ochrany vložení přístupových práv k souboru
  . nebo příkaz pro přidání přístupových práv pro čtení pro kohokoli v systému

* autorizované a neautorizované stavy matice ochrany
 - matice ochrany určuje, jaké operace proces v dané doméně může vykonat
 - co se stane, pokud se někomu podaří vykonat sekvenci příkazů, kterými dojde ke změně matice ochrany?
 - je zřejmé, že množina všech možných matic může být rozdělena do dvou podmnožin: množina autorizovaných stavů, množina neautorizovaných stavů

- otázka velké části výzkumu: máme-li počáteční autorizovaný stav a množinu příkazů, můžeme dokázat že se systém nikdy nedostane do neautorizovaného stavu?
- obecně obtížné (pro některé systémy nerozhodnutelné), pro konkrétní systémy je možné dokázat

* většina OS dovoluje uživatelům určit kdo může číst a modifikovat jejich soubory a další objekty
* v některých prostředích je třeba přísnější bezpečnost (vojenství, firemní výzkum, nemocnice, banky)
  - organizace stanoví pravidla, kdo co může vidět - pravidla nemohou být jednotlivými vojáky nebo lékaři atd. porušena
  - název vícestupňové modely bezpečnosti (multilevel security models)

* model Bell-La Padula (Bell & La Padula 1973)
  - původně navržen pro vojenské účely
  - každý dokument má přiřazen stupeň utajení (důvěrné, tajné, přísně tajné)
  - lidé mají také přiřazeny stupně podle typu dokumentů, které mohou vidět (např. generál může vidět všechny dokumenty, důstojník pouze důvěrné apod.)
  - procesy mají stupeň podle uživatelů kterým patří
  - model podle Bella a La Paduly má následující pravidla:
    . proces běžící na stupni K může číst pouze objekty na svém a nižších stupních
    . proces běžící na stupni K může zapisovat pouze objekty na na svém a vyšších stupních (důstojník může prozradit generálovi vše co ví, ale generál nemůže prozradit důstojníkovi vše co ví)

* problém s modelem: hlavní záměr je, aby systém uchoval tajemství
  - je zřejmé, že v mnoha praktických případech podstatné další vlastnosti => existují další modely

”Oranžová kniha”

* National Computer Security Center (součást NSA) - definovalo kritéria pro třídy bezpečnosti OS (1983 ”Trusted Computer System Evaluation Criteria” = TCSEC, nazýváno též ”the Orange Book”)
  - certifikovala se úplná konfigurace včetně HW
  - nahrazeno jinými standardy, ale stále užitečné pro základní rozlišení

* Orange Book rozděluje systémy do sedmi kategorií podle poskytované kontroly:
  - úroveň D: nejsou žádné požadavky na bezpečnost, nejsnáze dosažitelné (MS DOS, Windows 95/98/Me)
  - úroveň C pro prostředí s navzájem spolupracujícími uživateli
    . C1 - OS v chráněném režimu, autentizované přihlašování, možnost aby uživatelé specifikovali které soubory (zdroje) budou zpřístupněny ostatním uživatelům a jak (discretionary access control), požadavek minimálního testování a dokumentace (např. UNIX)
    . C2 - přidává požadavek ochrany přístupu na úrovni jednotlivých uživatelů, ochrana při znovupoužití objektu - uživatel nesmí mít možnost vidět data objektu, který jiný uživatel uvolnil (zrušil soubor, uvolnil paměť; všechny objekty jsou inicializovány před alokací uživateli), bezpečnostní audit - systém musí detekovat a zaznamenávat pokusy o vytvoření, přístup, zrušení zdroje; musí být možné zjistit, kdo se pokusil o neautorizovanou akci
    - B a A - uživatelům a objektům jsou přiřazena bezpečnostní návěští (např. ”neklasifikováno”, ”tajné”, ”přísně tajné”), systém musí umět model Bell-La Pandula
      . B1 = Labeled Security Protection, viz výše
      . B2 - systém musel být navržen shora dolů modulárním způsobem, návrh musí být prezentován tak aby ho bylo možno ověřit, analýza možných covert channels
      . B3 = B2 + ACL + bezpečnostní domény + formální TCB + bezpečné zotavení po havárii
    - A1 - formální model + důkaz jeho správnosti + demonstrace, že systém odpovídá modelu + formální analýza covert channels

* cílem je, aby jeden proces nemohl druhému předávat informace neřízeným

způsobem
- pomocí matice ochrany můžeme zajistit, aby nemohly komunikovat mechanismem zápis/čtení apod.
- covert channel např. bit 1 = server zatěžuje CPU, bit 0 = nezatěžuje CPU
- cílový proces může měřit čas odpovědi systému, případný šum může být eliminován pomocí samoopravných kódů
- hledání covert channels je značně obtížné

* TCSEC bylo nejznámější, ale dnes považováno za zastaralé
* následovník 1996 "Common Criteria for Information Technology Security Evaluation" (CCITSE; US, GB, Německo, Francie, Kanada, Holandsko)
* http://www.radium.ncsc.mil/tpep/library/ccitse
* obvykle nazýváno Common Criteria (CC), začíná být uznáváno
* nové systémy (Windows XP apod.) budou ověřovány už pomocí CC
* ověřování trvá několik let, bude ještě trvat
* není a nebude pravda "Windows 2000 supports C2-level security" (neplatí bez uvedení konfigurace, NCSC už neověřuje oproti TCSEC)

Criterion	D	C1	C2	B1	B2	B3	A1
Security policy							
Discretionary access control		X	X	→	→	X	→
Object reuse			X	→	→	→	→
Labels				X	X	→	→
Label integrity				X	→	→	→
Exportation of labeled information				X	→	→	→
Labeling human readable output				X	→	→	→
Mandatory access control				X	X	→	→
Subject sensitivity labels					X	→	→
Device labels					X	→	→
Accountability							
Identification and authentication		X	X	X	→	→	→
Audit			X	X	X	X	→
Trusted path					X	X	→
Assurance							
System architecture		X	X	X	X	X	→
System integrity		X	→	→	→	→	→
Security testing		X	X	X	X	X	X
Design specification and verification				X	X	X	X
Covert channel analysis					X	X	X
Trusted facility management					X	X	→
Configuration management					X	→	X
Trusted recovery						X	→
Trusted distribution							X
Documentation							
Security features user's guide		X	→	→	→	→	→
Trusted facility manual		X	X	X	X	X	→
Test documentation		X	→	→	X	→	X
Design documentation		X	→	X	X	X	X

* v tabulce jsou uvedeny úrovně bezpečnosti podle Orange Book:
- X = nový požadavek pro danou úroveň
- -> = požadavek z nižší kategorie

Poznámka (reakce na dotaz)

V systémech typu UNIX používají aplikace pro logování knihovní fce openlog(), syslog() a closelog(); tyto fce komunikují s daemonem syslogd(8), který se řídí konfiguračním souborem /etc/syslog.conf. V konfiguračním souboru je uvedeno, jak naložit s jednotlivými požadavky podle služby a priority požadavku (služba je např. cron, mail, security, daemon; priorita může být např. warning, err, emerg). Program může záznamy zapisovat do souboru (např. /var/log/auth.log), na terminál, případně posílat na jiný stroj.

[]

❇