Podstawowe zasady bezpieczeństwa

■ zasada nieuchronnego kompromisu:

> bezpieczeństwo jest odwrotnością wygody

■ możliwe podejścia do problemu zabezpieczenia:

- tworzenie zabezpieczeń tam gdzie są one niezbędne
- monitorowanie pracy systemu
- wykrywanie zaistniałych awarii i naprawianie uszkodzeń, a następnie śledzenie przyczyn awarii i ewentualne zatykanie dziur
- planowanie kryzysowe
- testowanie opracowanych systemów zabezpieczeń

■ inna niepodważalna zasada:

> system jest bezpieczny tylko w takim stopniu, jak jego najmniej bezpieczny element

Rodzaje zagrożeń bezpieczeństwa systemu komputerowego

- bezpieczeństwo fizyczne systemu: kradzież, zniszczenie, uszkodzenie, np. pożar, zalanie, skok napięcia
- bezpieczeństwo danych: uszkodzenie, sabotaż, kradzież, ujawnienie
- ciągłość dostępu do systemu: awarie sprzętu, oprogramowania, awarie zasilania, wentylacji, klimatyzacji, ogrzewania, ciągłości dostaw (np. paliwa do generatorów awaryjnych)

Zabezpieczenia istniejące w systemie Unix

■ zabezpieczenia standardowe:

- hasła bronią systemu przed dostępem osób niepowołanych
- prawa dostępu do plików zabezpieczają system przed niedozwolonymi poczynaniami użytkowników (błędy użytkowników, błędy oprogramowania, włamania)
- hasła i prawa dostępu do plików łącznie pozwalają również użytkownikom zabezpieczyć je przed samymi sobą i sobą nawzajem (podobnie: błędy lub zła wola)

■ inne zabezpieczenia:

- szyfrowanie zabezpiecza poufność danych

Ochrona kont i haseł

- zapewnienie haseł niepustych, nietrywialnych
- okresowe zmienianie haseł, „starzenie" haseł (ang. *password aging*)
- plik shadow
- niedopuszczenie do przechowywania haseł przez użytkowników „gołym tekstem", nieprzesyłanie ich przez sieć
- niedopuszczenie do współużywania jednego konta (i hasła) przez różnych ludzi
- używanie różnych haseł na różnych komputerach (zwłaszcza root)
- możliwość logowania się bez podania hasła (.rhosts, klucze)
- sieciowy system kont Yellow Pages/NIS
- inne, mocniejsze zabezpieczenia w razie potrzeby

Ochrona systemu plików

- programy *set-uid*
  - ochrona poprawnych programów *set-uid* (go-r)
  - wykrywanie dzikich programów *set-uid* (UWAGA: NFS)
  - zasady pisania programów *set-uid*
    - niedopuszczenie do wywołania interpretera poleceń: `system`, `popen`, `execlp`, `execvp`
    - odwoływanie się do bezwzględnych ścieżek do plików
    - tworzenie pseudoużytkowników gdy potrzebny jest dostęp do ograniczonych zasobów
- sprawdzanie plików i katalogów systemowych
  - prawa dostępu (/etc /dev /dev/dsk)
  - sumy kontrolne (/etc /sbin /bin /lib)
- sprawdzanie plików i katalogów użytkowników (niebezpieczne lub niezgodne z zasadami prawa dostępu)

Monitorowanie pracy systemu i wykrywanie awarii

- rejestry wydarzeń systemowych, przegląd rejestrów, syslog
- rejestr nieudanych połączeń (lokalnych i sieciowych) `/var/adm/loginlog`
- rejestr prób wejścia na konto root'a przez su `/var/adm/sulog`
- rejestr wszystkich wejść do systemu `/var/adm/wtmp` (`/var/adm/wtmpx`)
- wykrywanie niepoprawnych kont (puste hasła, UID=0)
- analiza bieżącej pracy systemu (`top`, `ps`)
- sumy kontrolne krytycznych plików systemowych
- wyszukiwanie plików *set-uid* 0, zabezpieczenie NFS
- system accounting (System V): rejestr wszystkich wywołań wszystkich programów
- czasy ostatniej modyfikacji i ostatniego dostępu do plików
- wydarzenia sieciowe

Usługi `r*`

Następujące polecenia pozwalają użytkownikowi posiadającemu konto na innej maszynie unixowej (lub Unixo-podobnej) na wykonywanie operacji bez konieczności logowania się, o ile zdalna maszyna realizuje odpowiednie usługi, i uważa maszynę lokalną za **równoważną** sobie:

rcp — kopiowanie plików

rlogin — włączanie się na zdalną maszynę

rsh — wykonywanie poleceń na zdalnej maszynie przez interpreter poleceń

Usługi te wykorzystują koncepcję równoważności maszyn (ang. *host equivalence*) zakładającą, że użytkownik prawidłowo wlogowany na jednej z równoważnych maszyn, może być wpuszczony na drugą bez autoryzacji.

Równoważność maszyn może zdefiniować administrator w pliku `/etc/hosts.equiv`. Może ją również zdefiniować użytkownik dla swojego konta za pomocą pliku `~/.rhosts`.

```
$HOME/.rhosts                   /etc/hosts.equiv
hostname   username             hostname
                                +         username
                                -hostname
```


Usługi `r*` — zagrożenia

Równoważność maszyn jest uznawana za niebezpieczną i niepożądaną własność, a ze względu na brak kontroli nad jej stosowaniem przez użytkowników, normalną praktyką jest wyłączanie usług `r*`, zwykle obsługiwanych przez `inetd`.

Jedną z przyczyn, dla których równoważność maszyn jest zagrożeniem, z którego administrator może nie zdawać sobie sprawy, jest niezrozumienie zasad interpretowania plików `.rhosts` i `/etc/hosts.equiv`:

- jeśli administrator wyłączy jakiś zdalny komputer lub użytkownika w pliku `/etc/hosts.equiv` to indywidualni użytkownicy i tak mogą udzielić im zezwoleń w swoich plikach `.rhosts`
- pozycja `hostname username` wpisana w pliku `/etc/hosts.equiv` jest interpretowana w ten sposób, że pozwala zdalnemu użytkownikowi `username` włączać się na konto **dowolnego** użytkownika lokalnego

Szyfrowane połączenia między systemami: `ssh`

Ze względu na te zagrożenia programy r* nie powinny być w ogóle używane. Jednak usługi takie są wygodne i potrzebne. W ich miejsce można stosować bezpieczniejsze programy eliminujące powyższe zagrożenia i stosujące szyfrowanie.

Programy `ssh` i `scp` można stosować jako bezpieczne zamienniki `rsh` i `rcp`, i można wręcz podłożyć programy s* w miejsce programów r* w systemie. W ten sposób efektywnie eliminujemy programy r* i ich odpowiadające usługi sieciowe, ale jednocześnie jakby „zachęcamy" użytkowników do stosowania bardziej bezpiecznych programów. Należy tylko pamiętać o wyłączeniu obsługi starych usług r* w systemie.

Usługi ssh/scp oparte są na szyfrowaniu kluczem publicznym. Wymagają one wygenerowania pary kluczy szyfrowania dla komputera, które potrzebne są do wstępnego nawiązywania połączenia. Ponadto, każdy użytkownik musi mieć wygenerowaną własną parę kluczy szyfrowania:

```
> ssh-keygen
> ls -l /home/witold/.ssh:
total 90
drwxr-xr-x   2 witold   gurus        512 gru 28 11:13 ./
drwxr-xr-x 115 witold   gurus      24576 mar 20 09:43 ../
-rw-------   1 witold   gurus        604 gru 28 11:12 authorized_keys
-rw-r--r--   1 witold   gurus         31 kwi 13  2005 config
-rw-------   1 witold   gurus        668 kwi 10  2005 id_dsa
-rw-r--r--   1 witold   gurus        603 kwi 10  2005 id_dsa.pub
-rw-------   1 witold   gurus        887 kwi 10  2005 id_rsa
-rw-r--r--   1 witold   gurus        223 kwi 10  2005 id_rsa.pub
-rw-------   1 witold   gurus        528 kwi 10  2005 identity
-rw-r--r--   1 witold   gurus        332 kwi 10  2005 identity.pub
-rw-------   1 witold   gurus       8228 mar 14 23:47 known_hosts
-rw-r--r--   1 witold   gurus        609 lis 11  2004 known_hosts2
-rw-------   1 witold   gurus       1024 kwi  5  2005 prng_seed
-rw-------   1 witold   gurus        512 paź 31  2001 random_seed
```


Nawiązywanie połączeń

Korzystanie z programów `ssh/scp` wymaga podawania hasła przy nawiązywaniu każdego połączenia:

```
shasta-3184> scp -p ~/mozilla.ps sierra:/tmp
witold@sierra's password:
mozilla.ps          100% |*****************************|   937 KB    00:01
```

Każdorazowe podawanie hasła jest często niewygodne i istnieją sposoby jego uniknięcia:

- `.shosts`
- `/etc/hosts.equiv` albo `/etc/shosts.equiv`
- instalacja kluczy na zdalnym komputerze: `$HOME/.ssh/authorized_keys`
- `ssh-agent`

Na szczęście administrator może kontrolować użycie dwóch pierwszych możliwości za pomocą systemowych plików konfiguracyjnych.

Tunelowanie połączeń przez `ssh`

Dodatkową przydatną własnością programu `ssh` jest zdolność **tunelowania połączeń**, czyli zainstalowania pary portów sieciowych na komputerach po obu stronach połączenia `ssh`, w celu uruchamiania przez użytkownika własnych programów komunikacyjnych, wykorzystujących bezpieczne, szyfrowane łącze `ssh` zamiast samodzielnego nawiązywania połączeń sieciowych.

Na przykład, jeśli chcemy (z jakichś powodów) zapewnić dostęp do zdalnego komputera przez program `ftp`, który jest przestarzałym protokołem komunikacyjnym wymagającym przesyłania przez sieć informacji autoryzacyjnej otwartym tekstem, to możemy zainstalować własny serwer `ftp` na tunelu `ssh`, i komunikować się w sposób bezpieczny. (Takie ćwiczenie będzie wykonywane na laboratorium.)

Innym przykładem zastosowania tuneli `ssh` jest przekazywanie połączeń X Window. Jest to przydatne nie tylko dla zabezpieczenia tych połączeń, ale często z jakichś powodów nie możemy na danym komputerze odbierać połączeń X Window na porcie 6000 (na przykład komputer jest za *firewall*-em, albo w sieci prywatnej z translacją adresów NAT).

Narzędzia przydatne do zapewnienia bezpieczeństwa systemu

- `md5sum` — obliczanie sygnatur plików
- `tripwire` — program do sprawdzania poprawności plików systemowych
- `crypt` — program do szyfrowania, głównie haseł
- `crack` — program do łamania haseł
- `cops` — program do kompleksowego sprawdzania systemu i wykrywania dziur w zabezpieczeniach