Linuxový víkend, 9. - 10. březen 2002, Praha

Alternativní bezpečnostní subsystémy pro Linux

Václav Lorenc

<xlorenc1@fi.muni.cz>

Obsah přednášky

Alternativní bezpečnostní subsystémy pro Linux

- slabiny stávajícího systému oprávnění,
- LIDS – Linux Intrusion Detection System,
- RSBAC – Rule Set Based Access Control,
- Medusa DS9 Security System,
- (SELinux – Security Enhanced Linux),

Stávající systém oprávnění

- možnost určit přístupová práva (pro čtení, zápis, provádění) k souboru pouze pro vlastníka, skupinu a pro ostatní,

  ```
  -r-s--x--x  1 root  root /usr/bin/passwd
  ```

  ⇒ nepříliš jemné rozlišení, lze si pomoci sdružováním uživatelů do různých skupin (problémy s počtem skupin a konzistencí oprávnění)

- méně propracované u sítě, IPC a procesů (nemají definovaná přístupová práva na základě vlastníka, skupiny a ostatních)

- procesy mají po spuštění přiřazeno číslo uživatele, pod kterým se provádí, obvykle shodné s tím, kdo program spustil, příp. je tu možnost zapůjčování UID/GID (speciální příznaky v atributech souboru)

- **ale hlavní problém** – super-uživatel smí vše! :-(

Jak to spravit?

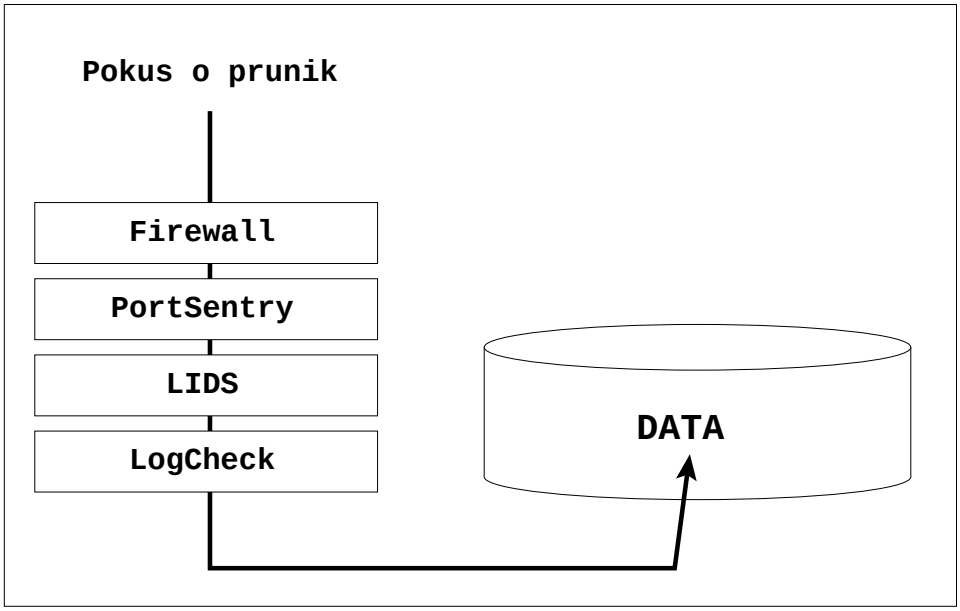
- zrušit superuživatele :-)
- ochrana na úrovni jádra
- omezit superuživatele – v linuxovém jádře podpora tzv. **capabilities**, rozdělení administrátorských práv do několika oblastí
  - souborový systém (změna vlastníka, přístupy)
  - procesy (zasílání signálů, změna uid/gid, změna capabilities)
  - síť (používání portů < 1024, RAW pakety, administrace sítě)
  - IPC
  - administrace systému (přidávání modulů do jádra, práva na restart/vypnutí systému, změna času...)
- problém s capabilites – nedostatek konfiguračních nástrojů
- další vylepšení – role, ACL (Access Control List)

$\Longrightarrow$ použití nějakého bezpečnostního subsystému

LIDS – Linux Intrusion Detection System

- autorem Xie Huagang (xie@gnuchina.org)
- patch do jádra (2.2.x, 2.4.x i 2.5.x) + administrátorské nástroje
- v současné době možnost používat i jako **LSM** (Loadable Security Modules)
- 3 základní typy úkolů
  - protection
  - detection
  - response
- ve světě velmi oblíbený $\Longrightarrow$ velká spousta dokumentace a příkladů

- časté použití:



LIDS – přehled možností

- nastavení capabilities pro celý systém
- nastavení capabilities pro jednotlivé procesy
- rozsah portů pro použití daným procesem
- přístupová práva jednotlivých procesů (např. login, sshd, su)
- časově omezená změna oprávnění (logrotate)
- **zapečetění systému**
- ochrana procesu před signály
- zákaz zápisu do souboru, příp. pouze přidávání
- skrytí procesu nebo souboru
- ochrana před zahlcením logu
- předávání oprávnění potomkům (možnost zvýšení bezpečnostního rizika)

LIDS – příklady

a) zákaz přístupu k souboru /etc/shadow

```
[root@mythago:~]# lidsconf -A -o /etc/shadow -j DENY
[root@mythago:~]# ls /etc/shadow
ls: /etc/shadow: No such file or directory
[root@mythago:~]# lidsconf -A -s /bin/login -o \
                  /etc/shadow -j READONLY
```

b) rozsah povolených portů

```
/sbin/lidsconf -A -s /bin/httpd \
        -o CAP_NET_BIND_SERVICE 80-80,443-443 -j GRANT
```

c) omezení zápisu pro passwd

```
lidsconf -A -s /usr/bin/passwd -o /etc  \
         -j WRITE
lidsconf -A -s /usr/bin/passwd -o /etc/hosts.allow \
         -j READONLY
lidsconf -A -s /usr/bin/passwd -o /etc/hosts.deny \
         -j READONLY
lidsconf -A -s /usr/bin/passwd -o /etc/rc0.d \
         -j READONLY
...
lidsconf -A -s /usr/bin/passwd -o /etc/init.d \
         -j READONLY
lidsconf -A -s /usr/bin/passwd -o /etc/cron.d \
         -j READONLY
lidsconf -A -s /usr/bin/passwd -o /etc/pam.d \
         -j READONLY
...
```

LIDS – závěr

- všichni vše nebo nic, nerozlišuje uživatele :(
- spousta vlastností bez ladu a skladu :(
- občas trochu sporné vlastnosti (schování procesu proces pouze schová, ale pokud někdo zná jeho pid, je pořád schopen mu poslat signál)
- podpora pro nejnovější jádra (i vývojová) :)
- přístup k souborům na základě inodů $\Longrightarrow$ nepříjemnosti např. s programem passwd, vzrůstá komplikovanost řešení
- **pro firewally a routery výhodné** (málo uživatelů, obvykle žádný aktivní krom správce) – malá režie, dobrá práce s logy, „zapečetění“ routovacích tabulek a firewallovacích pravidel

RSBAC – Rule Set Based Access Control

- autor Amon Ott, projekt původně diplomovou prací
- modulární systém s propracovanými bezpečnostními modely
- dokumentace (alespoň úvodní) je i v češtině, materiály na webu (vcelku dostatek, i s příklady)
- opět patch do jádra + nástroje pro administraci
  - konfigurace je dvojího druhu – „pohodlná“ (libdialogs) a „složitá“ (pomocí příkazového řádku)
  - výhody i nevýhody, převážně se projeví u většího množství konfigurovaných strojů
  - binární forma konfigurace :((riziko při poškození souborového systému)
- rozdělení pravomocí – system administrator, security officer, data officer

RSBAC – instalace

- zřídit uživatele s uid 400 a 401 (security officer, data officer)
- provést úpravu jádra, zaškrtnout bezpečností moduly, rozmyslet si způsob logování (syslog vs. rsbac)
- vytvořit si dvě jádra – jedno „ostré“, druhé pro napravování škod a další administraci
- parametry jádra
  - `rsbac_softmode` – „zkušební režim“
  - `rsbac_auth_enable_login` – povolení prvotního loginu
- přihlásit se a začít napravovat nefunkčnost způsobenou RSBAC

Přehled modulů

- nejzajímavější:
  - AUTH – všeobecné sledování procesů
    - *auth_may_setuid*
    - *auth_capabilities*
  - RC – Role Compatibility
  - MAC – Mandatory Access Control
  - FF – File Flags
    - *no_execute*, *secure_delete*, *search_only*, *write_only*, *execute_only*, *read_only*
    - dědičnost (+ masky), potlačení dědičnosti
  - ACL – Access Control List
  - Malware Scan :-)
  - **možnost napsat a přidat vlastní moduly**
- je dovoleno zapínat/vypínat za běhu systému (nutno povolit při kompilaci jádra)
- pro rozhodnutí o nepovolení přístupu k objektu stačí zamítnutí jednoho z aktivních modulů, každý dostane šanci se „vyjádřit“ $\Longrightarrow$ možnost kombinovat jednotlivé moduly a využívat jejich výhod

RSBAC – ACL požadavky

- *add_to_kernel* – pro moduly jádra, nahrání modulu do jádra
- *alter* – změna řídících údajů IPC
- *append_open*
- *change_group*, *change_owner*
- *chdir*, *close*, *create*, *delete*, *execute*, *rename*, *search*
- *get_status_data* – zjištění informací o souboru pomocí stat(2)
- *link_hard*, *modify_access_data*
- *modify_attribute*, *read_attribute* – změna, čtení atributů RSBAC
- *get_permissions_data* – zjištění UN*Xových práv
- *modify_permissions_data* – změna UN*Xových přístupových práv
- *modify_system_data* – změna systémových dat (porty, . . .)
- *mount*, *umount*
- *read*, *read_open*, *read_write_open*, *write*, *write_open*
- *remove_from_kernel* – odstranění modulu z jádra
- *send_signal* – pro programy, smí zasílat signály jiným procesům
- *switch_log* – změna úrovně logování RSBAC
- *switch_module* – aktivace/deaktivace jednotlivých modulů RSBAC
- *shutdown*, *terminate*, *trace*, *truncate*

RSBAC – RC (Role Compatibility)

- maximálně 64 rolí v systému
- vhodnost:
  - pro systémy, kde je možné jednoduše seskupit subjekty a objekty do typů a rolí
  - pokud je zapotřebí silné rozdělení pravomocí
  - je-li třeba kontrolovat přístup spíše podle procesů než uživatelů
- tři předdefinované role – General User, System Admin, Role Admin
- každý uživatel má roli, dědí se pro každý jeho proces
- změna role:
  - změnou vlastníka procesu
  - procesem na požádání („compatible“ roles)
  - možnost propůjčovat roli při spuštění procesu (password administrator)
- možnost nastavit stav nových objektů v systému, příp. zakázat jejich vytváření (*no_create*)

RSBAC – výpisy z logu

```
09:30:31 mythago PAM_unix[606]:
         (su) session opened for user root by valor
09:30:31 mythago kernel:
         rsbac_adf_request(): request CHANGE_OWNER,
         caller_pid 607, caller_prog_name su,
         caller_uid 500, target-type PROCESS, tid 607,
         attr owner, value 0, result NOT_GRANTED by AUTH
09:30:31 mythago PAM_unix[606]:
         (su) session closed for user root

                              ...

09:30:22 mythago xfs:
         xfs startup succeeded
09:30:22 mythago kernel:
         rsbac_adf_request(): request CHANGE_OWNER,
         caller_pid 559, caller_prog_name xfs,
         caller_uid 0, target-type PROCESS, tid 559,
         attr owner, value 43, result NOT_GRANTED by AUTH
09:30:22 mythago xfs:
         fatal: couldn't set userid to xfs user
```

RSBAC – ukázka různorodosti řešení (1)

Cíl administrátora:

ochránit všechny důležité spustitelné soubory před změnou

- společné kroky pro všechny moduly
  - identifikovat všechny adresáře obsahující soubory vyžadující ochranu plus všechny další samostatně ležící soubory (ty je možné najít i později metodou zjišťování, co nefunguje :-))
- řešení pomocí modulu **File Flags**
  - označeným adresářům přiřadit pouze práva *search_only* a *read_only*
  - nastavit příznak *read_only* všem samostatně ležícím spustitelným souborům, které chceme ochránit
  - je-li nutné číst obsah označených adresářů, je možno odebrat právo *search_only*
  - neobsahuje-li adresář žádné skripty a věci vyžadující čtení a interpretaci souboru, je možné přiřadit mu příznak *execute_only*
- řešení pomocí modulu **RC**
  - vytvořit nový FD, nazvaný např. „Executables“
  - všem rolím dát práva *search* a *execute* k tomuto typu
  - budou-li se vykonávat skripty, resp. je-li nutné tyto soubory otevírat, poté přidat ještě práva *read_open* a *close*
  - chcete-li, aby fungovalo doplňování v shellu, je nutné přidat i práva *read*, možná také *get_status_data* a *get_permissions_data*
  - všem označeným adresářům a souborům nastavit typ „Executable“

  - chcete-li přeci uživatele, který bude moci tyto soubory měnit, vytvořit další roli a přiřadit nezbytná práva typu „Executable“ pro přístup z této role

- řešení pomocí **ACL**
  - nastavit masku na práva *search* a *execute*
  - je-li potřeba otevírat nebo číst soubory (skripty), přidat i práva *read_open* a *close*
  - chcete-li, aby fungovalo doplňování v shellu, je nutné přidat i práva *read*, možná také *get_status_data* a *get_permissions_data*
  - **pozor!** právo *supervisor* zahrnuje ostatní práva a nemůže být maskou zrušeno! (defaultně pouze u uživatele Security Officer)
  - pro potřeby změny souborů je možné vytvořit další skupinu a přidat jí příslušná práva v ACL daného adresáře nebo souboru

RSBAC – ukázka různorodosti řešení (2)

Cíl administrátora:

zabránit vykonání nehlídaných souborů nebo knihoven

- kroky společné všem modulům
  - identifikovat všechny adresáře obsahující důležité spustitelné soubory + adresáře s dynamickými knihovnami (*.so)
  - opět je možné zjišťovat nedostatky a zbylé adresáře za běhu, ovšem pozor – opomenutí nějakého důležitého adresáře může způsobit neschopnost spustit většinu souborů na systému!
- řešení pomocí modulu **FF**
  - ze všech vybraných adresářů odstranit příznak *add_inherited*
  - příznak *add_inherited* odstranit i z ostatních souborů a knihoven
  - na kořenový adresář použít příznak *no_execute*
  - that's all :) s příznakem *add_ihnerited* všechny ostatní adresáře a soubory zdědí „schopnost“ nevykonávat soubory
- řešení za pomoci rolí
  - vytvořit nové FD, nazvané např. „Executables“ a „Libraries“
  - všem rolím dát práva *search* a *execute* k tomuto typu
  - budou-li se vykonávat skripty, resp. je-li nutné tyto soubory otevírat, poté přidat ještě práva *read_open* a *close*
  - chcete-li, aby fungovalo doplňování v shellu, je nutné přidat i práva *read*, možná také *get_status_data* a *get_permissions_data*
  - všem označeným adresářům a souborům nastavit typ „Executable“, adresářům s knihovnami a knihovnám typ „Libraries“

  - odebrat oprávnění *execute* u všech typů (krom našich dvou) všem rolím, doporučeno administrátorovi odebírat až poslednímu a předtím otestovat chování systému

- řešení za pomoci ACL
  - přidělit práva *search* a *execute* skupině 0 („Everyone“) všem dříve vybraným adresářům a souborům
  - opět příp. přidat i práva *read_open* a *close*
  - odstranit právo *execute* z kořenového adresáře, příp. ze všech položek standardního typu FD
  - máte-li u nějakých jiných adresářů či souborů přidělená speciální práva, odstraňte z nich *execute*. Zjištění všech položek ACL je možné pomocí `acl_tlist -r`.
  - opět platí, že subjekty mající právo *supervisor* mohou vše (a opět se to obvykle vztahuje pouze na uživatele Security Officer)

- proti LIDSu umí brát v úvahu jednotlivé uživatele
- hezká, přehledná konfigurace (nemusí být vždy výhodou)
- již od počátku práce s RSBAC je systém podstatně bezpečnější, díky „nápravě chyb“ je administrátor proveden celým systémem a jeho bezpečnostními riziky
- při vhodném postupu není možné RSBAC vypnout
- pomalejší, vyšší režie systému :-(
- změna bezpečnostních pravidel za chodu se chová spíše nedefinovaně
- nevýhoda binární podoby konfigurace v `/rsbac`

Výhledy do budoucna

- zrychlení
- možnost přesměrování požadavků na soubor/adresář jinam
- zlepšit chování při poškození informací v adresáři `/rsbac`
- dokončit správu uživatelů a hesel v modulu AUTH
- dále zlepšovat bezpečnost Linuxu, zejména v oblasti internetových serverů
- splnit požadavky k dosažení stupně bezpečnosti B1

Medusa DS9 Security System

- slovenský projekt, autoři Marek Zelem, Milan Pikula a Martin Očkajak
- patch do jádra + user-space autorizační server (constable)
- dokumentace – v angličtině jí moc není, ve slovenštině je možné sehnat diplomové práce autorů i práce přímo týkající se pouze Medusy
- počáteční systém téměř roven původnímu, žádné zabezpečení s dodanou defaultní konfigurací
- ukázkové příklady, škola hrou :-)
- konfigurační soubor nesoucí rysy programu v jazyce C $\Longrightarrow$ velká volnost při nastavování zabezpečení

Medusa DS9 – základní bezpečnostní rysy

- virtuální prostory (virtual spaces) – oddělené procesy nebo skupiny procesů, určení přístupových práv mezi VS
- sledované akce u souborů: access, create, delete, rename, link, execute
- sledované akce u procesů: fork, exec, send signal execute set uid program, set{re,s}uid, capability check, ptrace
- možnost sledovat a reagovat na libovolné volání služby jádra (syscall)
- rozhodnutí o povolení/zakázání přístupu provádí autorizační server (v současné době Constable) $\Longrightarrow$ větší možnosti při rozhodování
- o každém procesu stejné informace jako jádro + některé další, většinu je možné měnit
- postup při ověřování požadavku
  - otestování virtuálních prostorů
  - zjištění potvrzení operace
  - ověření autorizačním serverem
  - ověření systémovými právy
  - provedení operace
- typy odpovědí autorizačního serveru:
  - OK
  - SKIP
  - YES
  - NO

Medusa DS9 – příklad konfigurace (1)

```
for exec "/sbin/lsmod" {
        // Všem, kromě roota...
        if (euid != 0) {
                redirect "/usr/games/fortune";
        }

                        ...

[root@mythago:~]# lsmod
Module                  Size  Used by
parport_pc             25728   1 (autoclean)
lp                      6208   0 (autoclean)
parport                26400   1 (autoclean) [parport_pc lp]

[root@mythago:~]# su - valor

mythago:~$ /sbin/lsmod
Experience is that marvelous thing that enables you
recognize a mistake when you make it again.
                -- Franklin P. Jones
```

Medusa DS9 – příklad konfigurace (2)

```
for exec "/usr/sbin/in.telnetd" {
  // Indicator, this process can't run suid
  // program
  flags = 1;
  // monitor, when he tries to run suid program
  procact = P_SEXEC;
}
// when sexec event appears
on sexec {
  // is it disallowed process ?
  if (flags == 1) {
    // do not allow to run set uid program
    answer = NO;
  }
```

Medusa DS9 – závěr

- velká kontrola nad systémem, variabilita řešení
- dobrá slovenská dokumentace
- zpočátku špatná orientace v tom, kde začít
- při nevhodně zvoleném logování událostí možnost zahlcení
- do budoucna – nový constable, přepracovaný způsob konfigurace (mělo by být ovšem možné používat i starý contable)

SELinux – Security Enhnanced Linux

- z dílny americké NSA (National Security Agency), předchůdcem byl projekt Flask Security Architecture
- původně patch do jádra, v současnosti jako LSM
- user-space autorizační server + sada modifikovaných běžně používaných programů (`ps`, `ls`, `login`)
- chování při změně bezpečnostní politiky promyšlené do detailů
  - je možné měnit pravidla a práva za chodu s okamžitým promítnutím do stavu běžících aplikací,
  - při vytváření nových objektů (entit) se jim automaticky při vytvoření dají atomickou operací defaultní práva,
  - kontrola nejen přístupu do souboru, ale dokonce i předávání deskriptorů na soubory
- nové API v jádře, je možné vytvářet programy pracující i s bezpečnostním kontextem aplikace $\Longrightarrow$ xterm nepovolující cut-n-paste textu z terminálu roota do terminálu uživatele, ...
- konfigurační skript pro prvotní nastavení bezpečnostních kontextů, vhodné zejména pro RH (což však nevylučuje ostatní distribuce)
- textová konfigurace v několika souborech $\Longrightarrow$ výhoda při přenášení bezpečnostních pravidel na jiný stroj nebo i více strojů, poměrně nepřehledná :-(

Literatura

[1] Domovská stránka projektu RSBAC.
http://www.rsbac.de/.

[2] Linux Intrusion Detection System.
http://www.lids.org/.

[3] Medusa DS9 Security System.
http://medusa.fornax.sk/.

[4] Openwall Project.
http://www.openwall.com/.

[5] Security Enhanced Linux.
http://nsa.gov/selinux/.

[6] Stanislav Ievlev, Hynek Mlnarik; RSBAC pro začátečníky.
http://cesdis.gsfc.nasa.gov/beowulf/.