# Visão geral do Linux/Unix

# e

# Controle de Acesso

Victor Francisco Fonte

Universidade do Minho

vff@di.uminho.pt

2015

# Linux 101

- Multiprograma, multitarefa, multiusuário Kernel, bibliotecas, chamadas de sistema,
- processos Super-usuário, usuário regular Shell, sistema e linha de comando em nível
- de usuário programas Configurações e
- gerenciamento de serviços Interface gráfica e aplicativos gráficos opcionais
-
-

# Linux 101 (2)

- Programa versus processo Memória
- virtual com espaço de endereço isolamento Sistema de arquivos
- Comunicação entre processos
-

# Processos

- Programa em execução Espaço de
- endereço isolado Executado sob um
- ID de usuário e grupo
  - usuários reais ou virtuais IDs reais ou eficazes do usuário e do grupo
  -
- Pode gerar outros processos

# Processo: espaço de endereço

- Modelo linear tradicional Código (inferior), dados e pilha (superior) Dados: inicializados versus não inicializados (por exemplo, heap)
- O acesso ao endereço nulo (zero) é ilegal
-

# Processo: descritores de arquivo

- O descritor de arquivo representa um nível de kernel objeto semelhante a arquivo Processos descritores de arquivo herdados: entrada
- padrão, saída e erro Os processos herdam todos os descritores de arquivo abertos de seus pais
-

# Processo: criação e execução

- Os processos são criados por clonagem
  - espaço de endereço é copiado (preguiçoso) descritores de arquivo
  - abertos são herdados
- Programas são executados
  - espaço de endereço é substituído pelo conteúdo de um arquivo executável (ABI) descritores de arquivos abertos são
  - preservados

# Interprocessos Comunicação

- Arquivos IPC
- tradicional:
  - memória compartilhada
  - filas de mensagens
  - semáforos
- Pipes (nomeados ou sem nome), soquetes (local ou domínios de rede),...

# Arquivo e diretório Unix Permissões e Modos

- Sistemas compatíveis com POSIX
- POSIX 1:2008 também conhecido por Single Unix Especificação (SUS) V4

# Usuários, Grupos e Senhas

- ID do usuário (UID), ID do grupo (GID)
- Grupos primários e secundários DB do usuário: /etc/passwd (e /etc/shadow)
- Grupo DB: /etc/group
-
-

# Controle de acesso a Recursos

- Homogeneidade, simplicidade
  Arquivos, diretórios e outros recursos
- Propriedade e permissões
-

# Propriedade

- Proprietário do usuário atribuído Proprietário do
- grupo atribuído

# Classes de usuários

1. Proprietário do usuário
2. Membro do grupo proprietário
3. Outros (todos os outros)

Verificado por este pedido.

# Permissões

- Ler, escrever, executar Para cada classe de usuário Permissões
- definidas de forma independente
- Nem todas as combinações fazem sentido Arquivos e diretórios são semelhantes Mas semântica ligeiramente diferente
-
-

# Permissões

- umask: restrição padrão do usuário em permissões Permissões de 12 bits mantidas no
- resource i-node As permissões influenciam syscalls (não comandos) Somente super-usuário pode modificar a propriedade Somente o proprietário (ou super-usuário) pode modificar as perms
-

# Permissões definidas em Arquivos

- Lido: acessar conteúdo
- Escrever: modificar conteúdo Executar: executar conteúdo
-

# Diretórios

- Diretórios são arquivos especiais Mapear nomes para i-nodes Acesso a nomes protegidos por 'leitura' Acesso a i-nodes protegidos por 'execute'
-
-

# Permissões definidas em Dirs

- Lido: visualizar/listar nomes
  Escrever:
- adicionar/excluir/renomear nomes
  Executar: chdir, acessar i-nodes
- (stat)

- Executar permissão de pesquisa a.k.a
  Executar permissões necessárias ao longo
- de um caminho Permissões não herdadas
  do diretório pai
-

# Permissões e outras Atributos

- Permis, U-proprietário, G-proprietário, c/m-
- data, nome,... perms: tipo, perms, set
- uid/gid, pegajoso,... ex: -rwxr—r—, drwxr-
- xr-x ex: drwsrwsrwt, drwsrwt ex: -rwxrw-
- r—+

# Permissões: Experimentação

- chmod, chown, chgrp read (), write
- (), exec (), unlink (), stat () Straace
-

# Permissões: Experimentação

- mkdir adir && toque
- adir/afile is adir; ls -l adir; ls -l
- adir/afile chmod -x adir is
- adir; ls -l adir; ls -l adir/afile

# Usuário Real versus Efetivo IDs de & Grupo

- Padrão: (EUID, EGID) = (RUID, RGID) Se SUID, SGID são definidos em recursos
- EUID = ID do proprietário do usuário do recurso EGID = ID do proprietário do grupo de recursos

# Atrens SUID & SGID

- Útil, mas perigoso Viola o princípio de
- privilégios mínimos Procurar por SUID raiz
- Em particular, a permissão raiz SUID +
- 'write' Executável/não executável: 's'/ 'S'
-

# SUID & SGID em arquivos

- 'S no ID do proprietário do usuário: sem significado
- 'S no ID do proprietário do grupo: bloqueio obrigatório
- 's' não tem efeito sobre scripts... por quê?

# SUID e SGID em DLLs

- Objetos compartilhados: .so, .so. número
- Padrões em todo o sistema: /etc...
- Substituir: LD_LIBRARY_PATH Perigoso
- para SUID, SGID... Por quê? Se SUID ou
- SGID então LD_LIBRARY_PATH é ignorado

# Bit adesivo em arquivos

- Bit de texto pegajoso também conhecido por Código mantido em troca
- ou memória Descontinuado em favor da memória virtual Executável/não executável: 'T'/não' Nenhum efeito em não-executáveis
-

# SUID & SGID em Dirs

- SUID sem efeito em diretórios No Linux e no Solaris SGID: proprietário do grupo
- copiado para entradas (sem herança!) A semântica SGID não está no SUS V4
-

# Bit pegajoso em Dirs

- Se definido, apenas proprietário de um recurso, o proprietário do diretório (ou o super-usuário) pode mover/renomear/excluir o recurso Funciona em conjunto com 'write'
- perm

# Listas de Controle de Acesso

- Conjunto de permissões para usuários e grupos específicos Complementa o
- mecanismo UGO/RWX ACL padrão para diretórios pode ser herdada, mas geralmente são apenas copiados para entradas Proposta POSIX retirada Solução para o problema umask por diretório
-
-

# ACLs: Experimentação

- Serviços públicos: setfacl, getfacl
- Máscara de permissão efetiva versus real Denotado por um '+' após perms
-

# Adicional/Estendido Atributos

- Atributos adicionais:
  - NTFS, ex-família,... ext2: lsattr, chattr
  -
- Atributos estendidos:
  - nome, pares de valores getfattr,
  - setfattr
- Um '' ou '.' após perms, a menos que a ACL esteja definida

# Poderes enraizados

- Algumas operações ignoram permissões Verifique quem faz a solicitação: EUID = 0?
- Por exemplo, desligamento do sistema, ligação da porta,... Soluções: mecanismo de capacidades Programa começar como raiz, mas desistir recursos que eles não precisam Minimizar os privilégios a serem explorados
-

# Prisão Chroot

- altera o diretório raiz aparente afeta o
- processo de execução e seus filhos
- operação restrita ao superusuário uso: teste
- e desenvolvimento, dependência controle e compatibilidade, recuperação, privilégio separação

# Práticas recomendadas do Chroot

- mudar diretório de trabalho para a prisão antes chroot alterar o ID de usuário real/efetivo para um não-root manter
- o mínimo possível dentro da prisão têm root próprios tantos arquivos somente leitura presos quanto possível limitar todas
- as permissões de arquivos e diretórios criar um script de configuração de permissões chroot de dentro do próprio daemon (evite embrulhar)
-
-
-

# Práticas recomendadas do Chroot

- pré-carregar objetos carregados dinamicamente evite usar o arquivo /etc/passwd preso fechar descritores de arquivo
- agressivamente antes de chrooting arquivos de configuração de link do lado de fora atualizar variáveis de
- ambiente para refletir a nova raiz
-
-

# Chroot mínimo

- close (descritores de arquivo não utilizados );
- chdir (“prisão”);
- chroot (” . “); setuid
- (não-root-uid);

# Chroot

- chamadas relacionadas ao sistema:
- chroot () comandos relacionados: chroot
- precursor da prisão, Solaris Containers, Linux Contêineres, Docker, espaços de usuário Linux (kernel 3.8+)

# Leitura adicional

- http://en.wikipedia.org/wiki/Unix_security
- http://www.tldp.org/LDP/intro-linux/html/sect_03_04.html
- http://www.cse.psu.edu/~trj1/cse497b-s07/slides/cse497b-lecture-18-unixsecurity.pdf
- _____

_____