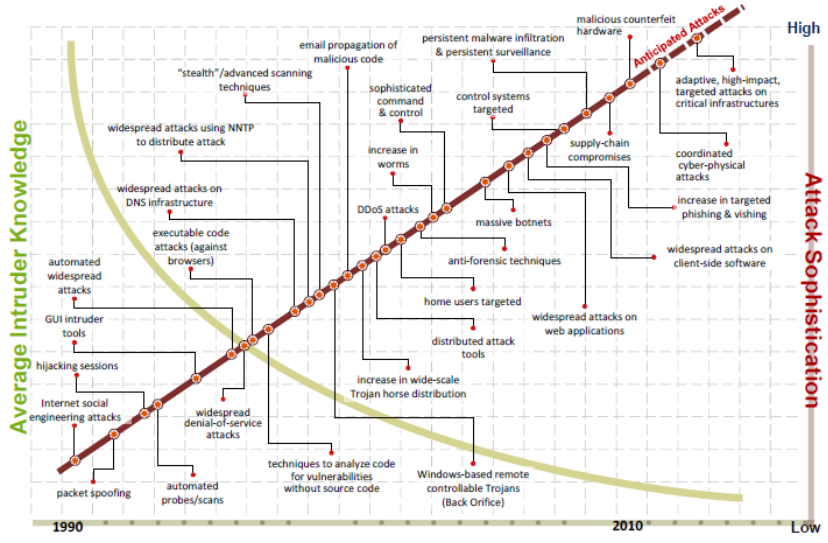
Sistemas de Detecção de Intrusões

Nuno Ferreira Neves

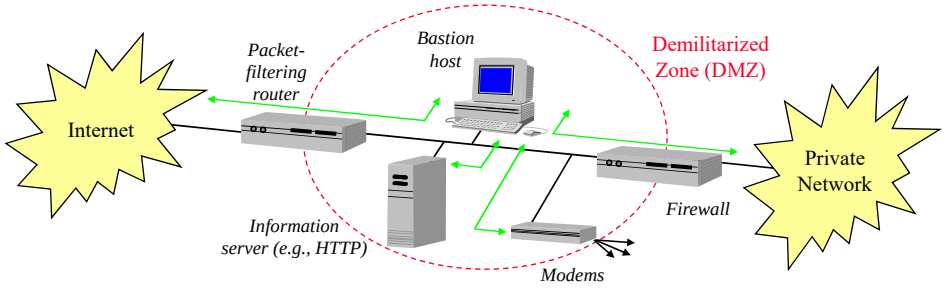
Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa

Sofisticação do Ataque – Sistemas de controlo



(Fonte: CERT)

Arquitectura de Segurança Tradicional



- Packet filtering router : tem como função básica a detecção de ataques de personificação de endereço (e.g., remover pacotes que chegam da Internet com endereços da rede privada, ou vice versa), e potencialmente alguns ataques específicos (e.g., *SYN flood*, e *Ping of Death*); normalmente, ***não*** evita ataques ao nível do protocolo aplicacional (e.g., bug num CGI)
- Bastion host : permite, por exemplo, a execução de formas mais seguras de autenticação
- Firewall : limita o tráfego entre a DMZ e a rede privada (e.g., só permite acessos de *login* a máquinas na rede privada se eles forem provenientes do *bastion host*; o servidor HTTP só pode contactar um servidor específico na rede privada)

Problemas com a Arquitectura Tradicional

- Qualquer um dos **componentes de software executados na DMZ pode ter um bug** de segurança - no router, sistemas de informação, sistema operativo
  - o servidor Web (o *Microsoft Internet Information Server* tinha um crash para certos tamanhos de URL)
  - um *script* executado pelo servidor (e.g., não testavam os delimitadores de comandos que recebiam dos utilizadores, e por isso permitiam a execução dum qualquer comando no sistema)
  - sistema de autenticação (o *rlogin* do AIX permitia o acesso a *root* de utilizadores remotos; o protocolo do Kerberos tem potencialmente alguns problemas)
  - um problema no protocolo que liga um servidor na DMZ e um servidor interno
- Um ou mais dos **mecanismos de segurança não está configurado** corretamente
  - existem estudos que mostram que uma percentagem muito elevada dos servidores Web são suscetíveis a alguma forma de ataque

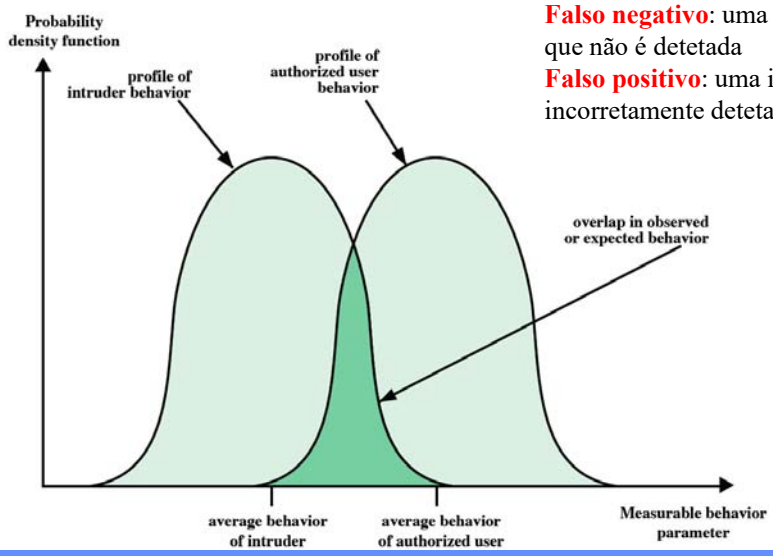
Detecção de Intrusões

- Um ***Sistema de Detecção de Intrusões (SDI)*** tem como principal função a observação e análise das actividades que ocorrem no sistema - rede de computadores, com o objectivo da detecção intrusões
- A intrusão pode ser proveniente dum
  - **utilizador interno** que consegue escalar/aumentar os seus privilégios, passando a ter acesso a recursos que anteriormente lhe eram negados
  - **intruso externo**, sem qualquer área no sistema, que consegue obter acesso a um dos computadores (com mais ou menos privilégios)

Principais componentes

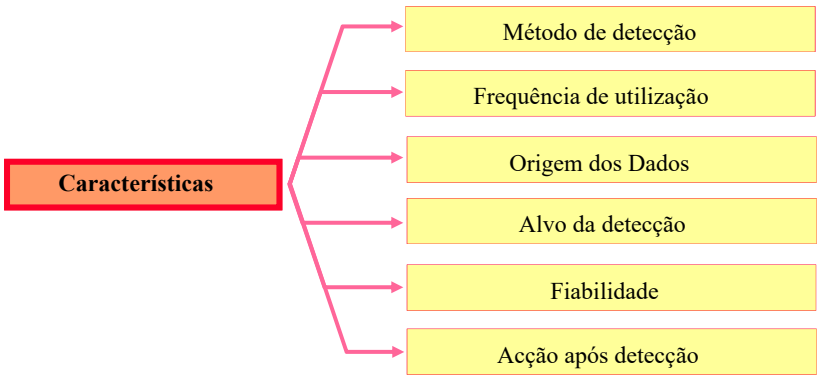
- **Sensores**
  - responsáveis por recolherem informação
  - qualquer parte do sistema pode ser monitorizada, e.g., acessos ao servidor Web ou certos troços da rede
  - enviam os dados para um analisador
- **Analisador**
  - recebe dados de um ou mais sensores ou de outros analisadores
  - determina se ocorreu uma intrusão correlacionando a informação recebida
  - poderá indicar ações que deverão ser tomadas para corrigir o problema
- **Interface com o utilizador**
  - fornece informação sobre os alarmes gerados e os eventos observados
  - pode indicar outra informação variada, como fornecer uma estimativa de risco do sistema que está a ser monitorizado
- **Arquivo de eventos**
  - suporte de auditoria e conformidade com certos standards/regulações

Principal dificuldade



Falso negativo: uma intrusão que não é detetada
Falso positivo: uma intrusão incorretamente detetada

Características Fundamentais do SDI



Método de deteção: Deteção baseada no **conhecimento**

- Compara as atividades observadas com uma lista de padrões/assinaturas de ataque
- Um padrão pode ser composto por um evento, sequências de eventos, números máximos de eventos ou expressões regulares (com E e OU) de eventos
- Assume que tudo o que **não é conhecido é válido**, o que significa que se a base de dados de padrões não for atualizada, muitos ataques podem não ser detetados
- **Baixo nível de falsos alarmes** (*falsos positivos*)

Exemplo de Padrão : Ataque *Land*

- O ataque *land* é um ataque de negação de serviço (*denial-of-service*) em que um pacote IP é enviado com endereços de emissão e recepção iguais; alguns sistemas operativos não sabiam como tratar este comportamento e paravam

Exemplo: Filtro programado com N-Code
(linguagem do *Network Flight Recorder* da Cisco)

```
filter pptp ip() {
# If the source Internet address is the same as the destination Internet address, then
    if ( ip.src == ip.des ) {
# Record the time the event happened, the Media Access Controller (MAC) address,
# the IP address of the source and destination computers
        record system.time, eth.src, ip.src, eth.dst, ip.dest to land_recorder;
    }
}
```

Algumas dificuldades na concretização dos SDI

- Definição de padrões genéricos que detetem pequenas variantes dos ataques, mas que ao mesmo tempo não considerem ações corretas como maliciosas
- Para padrões complexos que requerem a observação de vários eventos
  - armazenar informação que descreve o que foi visto no passado (i.e., a situação atual do potencial ataque)
  - libertar a informação relativa a potenciais ataques que acabam por nunca se concretizar
- Desenvolvimento de componentes capazes de detetar padrões no meio de muitos dados de uma forma eficiente (e.g., máquinas de estado, árvores de decisão, etc)
  - mesmo quando este objetivo é atingido, podem haver sempre situações em que existem mais dados do que a máquina onde o DSI está instalado pode processar; isto leva a que alguns dados não sejam analisados ...

Método de Deteção: Deteção baseada no **comportamento**

- Procura desvios de comportamento dos utilizadores, grupos de utilizadores, servidores, ..., face ao **comportamento *normal***
- Um **modelo** é associado a cada utilizador, cuja forma é determinada através da observação do comportamento do utilizador ao longo do tempo, recolhendo estatísticas relevantes (e.g., número de ficheiros abertos por semana ou número de tentativas a fazer *su root*)
  - normalmente existe um **período de aprendizagem** (e.g., um mês) onde os valores das variáveis são obtidos e o modelo é construído
  - depois o SDI **entra em funcionamento**, permitindo a gradual atualização das variáveis ao longo do tempo
- Ao longo dos anos várias técnicas têm sido propostas, incluindo métodos de aprendizagem automática
- Assume que os desvios são suspeitos, o que significa que alterações de comportamento momentâneas resultam em falsos alarmes
- **Potencialmente um elevado nível de falsos alarmes** (*falsos positivos*)

Conhecimento *vs.* Comportamento

- Vantagens do conhecimento
  - o número e tipo de eventos a considerar está limitado às atividades descritas nos padrões, o que potencialmente permite uma diminuição dos eventos a monitorizar
  - os cálculos tendem a ser mais eficientes do que com a deteção baseada no comportamento porque não usa virgula flutuante

- Desvantagens do conhecimento
  - escalabilidade e desempenho dependem do tamanho e arquitetura da base de dados de padrões
  - quando surgem novos tipos de ataques, é necessário adicionar novos padrões
  - a conversão de uma descrição de um ataque em linguagem natural para um padrão pode requer um esforço manual substancial
  - adição de padrões poderá ser difícil se não houver uma linguagem standard de especificação
  - não existe capacidade para se aprenderem novos padrões automaticamente

*(**até recentemente**, na prática, este método era o **mais usado** nos sistemas disponíveis para venda)*

Conhecimento *vs.* Comportamento

- Vantagens do comportamento
  - **permite detetar ataques *zero-day***
  - podem-se usar técnicas estatísticas e de aprendizagem automática bem conhecidas
  - como é necessário manter informação apenas sobre um conjunto de variáveis, não existe um consumo muito grande de memória
  - alguns dos comportamentos medidos são facilmente percebidos pelos operadores (e.g., número de tentativas de *login* falhadas)

- Desvantagens do comportamento
  - as hipóteses sobre os dados recolhidos podem não ser válidas sobre o ponto de vista estatístico (e.g., duas variáveis estão relacionadas em vez de serem independentes)
  - o comportamento de certos utilizadores pode não ser consistente ao longo do tempo
  - um adversário que compreenda o que está a ser medido, pode alterar lentamente o seu comportamento de forma que não seja detetado, ou pode usar ataques distintos
  - poderá não existir possibilidade de ordenar os eventos no tempo

Frequência de Utilização

- Um SDI pode estar constantemente a monitorizar o sistema de forma a que os ataques sejam detetados em tempo real, ***ou*** ser executado esporadicamente quando existe suspeita de intrusão
- É de notar que existe uma complementaridade entre os dois tipos de SDI

- **Detetores de vulnerabilidades**
  - periodicamente percorre o sistema à procura de vulnerabilidade
  - Como é que melhoram a segurança ?
    - » evitam as intrusões ao indicarem os problemas de segurança no sistema
    - » **detetam vulnerabilidades introduzidas pelo adversário após a intrusão**
  - têm as vantagens que consomem menos recursos - apenas quando correm, e que não requerem outras fontes de dados (e.g., *log* de auditoria de sistema)
  - têm as desvantagens que não detetam os ataques/intrusões em tempo real, e que um adversário que conheça a vulnerabilidades testadas pode evitar a sua deteção

Detectores de Vulnerabilidades (cont)

- As vulnerabilidades podem ser detetadas porque
  - uma versão do programa X tem um bug de segurança bem conhecido
  - a instalação por omissão do programa Y deixa um erro de configuração (e.g., deixa um ficheiro com demasiados privilégios de acesso)
  - um erro de administração deixa o sistema/aplicação num estado inseguro
  - alguém colocou no sistema um programa que é utilizado para ataques
- Os detetores podem ser
  - **locais** : o detetor deteta problemas no sistema onde se está a executar
    - » exemplos : último *patch* foi adicionado; ficheiros *.rhosts*; *buffer overflow*
  - **remoto** : o detetor procura problemas de segurança nos sistemas ligados à rede; esta atividade é conseguida através do envio de pacotes para os serviços de rede
    - » exemplos : *spoofed packets* para testar relações de confiança; *buffer overflow;* testar configuração da *firewall*;
  - Os detetores locais e remotos **complementam-se** um ao outro; existem também vulnerabilidades que podem ser detetadas por ambos, o também é útil

Exemplo : COPS

- O *Computer Oracle and Password Security System (COPS)* é estruturado como um conjunto de pequenos programas, cada um verificando alguma parte específica de segurança (e.g., utilizadores com *password* igual ao *username*)

Relatório após uma inspecção

```
ATTENTION:
Security Report for Thu Jun 1 20:04:20 WEST 2000
from host lolita.di.fc.ul.pt


**** root.chk ****
Warning! /etc/ftpusers exists and root is not in it
**** dev.chk ****
Warning!  /dev/fd0 is _World_ readable!
**** is_able.chk ****
Warning!  /etc/security is _World_ readable!
**** rc.chk ****
**** cron.chk ****
**** group.chk ****
**** home.chk ****
**** passwd.chk ****
**** user.chk ****
**** misc.chk ****
**** ftp.chk ****
Warning! root should be in /etc/ftpusers!
**** pass.chk ****
**** kuang ****
**** bug.chk ****
```


Origem dos Dados

- Rede
  - obtém-se informação sobre os cabeçalhos dos pacotes que transitam pela rede

    Exemplos : endereços IP, portos, números de sequência, tipo de protocolo, *checksums* ...

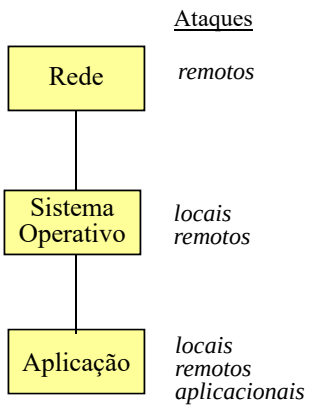
    Ataques : *address spoofing*
- Sistema Operativo
  - obtém informação sobre as actividades que transitam pelo sistema operativo

    Exemplos : quem abriu/apagou/criou um ficheiro

    Ataques : substituição do comando *ls*
- Aplicação
  - recolhe informação específica à aplicação

    Exemplo : dados sobre um *login* falhado; os logs de uma firewall ou de um router

    Ataques : alterar um *record* protegido da BD



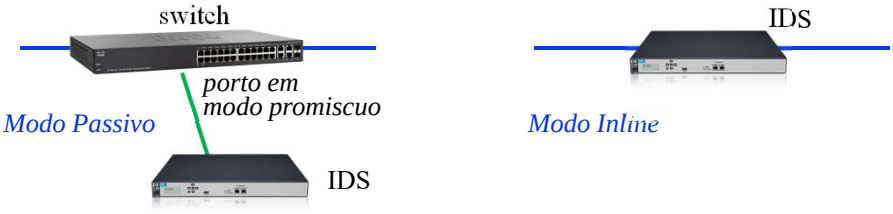
Exemplo: Auditoria de Eventos do SO

- Quase todos os SO comerciais têm um sistema de auditoria que permite a obtenção detalhada da maior parte das atividades no sistema
- Geralmente os eventos são ao nível das chamadas de sistema, e o administrador pode escolher que tipos de eventos e que objetos do sistema devem ser considerados pelo sistema de auditoria
- Alguma da **informação guardada por evento**
  - identificador do evento
  - detalhes sobre quem executa a operação (ids do utilizador, pids do processo)
  - *path* completo do programa que gerou o evento
  - instante em que ocorreu o evento
- Alternativas **no tipo de ações/dados em que se baseia**
  - tipo e ordem de chamadas a funções do sistema operativo >>>
  - info de auditoria recolhida pelo SO, armazenada localmente ou num servidor remoto
  - *checksums* de ficheiros, mas é necessário proteger os *checksums* de ataques
  - alterações no *registry* do windows

Chamadas a funções monitorizadas no Ubuntu Linux

accept, access, acct, adjtime, aiocancel, aioread, aiowait, aiowrite, alarm, async_daemon, auditsys, bind, chdir, chmod, chown, chroot, close, connect, creat, dup, dup2, execv, execve, exit, exportfs, fchdir, fchmod, fchown, fchroot, fcntl, flock, **fork**, fpathconf, fstat, fstat, fstatfs, fsync, ftime, ftruncate, getdents, getdirentries, getdomainname, getdopt, getdtablesize, getfh, getgid, getgroups, gethostid, gethostname, getitimer, getmsg, getpagesize, getpeername, getpgrp, getpid, getpriority, getrlimit, getrusage, getsockname, getsockopt, **gettimeofday**, getuid, gtty, ioctl, kill, killpg, link, listen, lseek, lstat, madvise, mctl, mincore, mkdir, mknod, mmap, mount, mount, mprotect, mpxchan, msgsys, msync, munmap, nfs_mount, nfssvc, nice, open, pathconf, pause, pcfs_mount, phys, pipe, poll, profil, ptrace, putmsg, quota, quotactl, read, readlink, readv, reboot, recv, recvfrom, recvmsg, rename, resuba, rfssys, rmdir, sbreak, sbrk, select, semsys, send, sendmsg, sendto, setdomainname, setdopt, setgid, setgroups, sethostid, sethostname, setitimer, setpgid, setpgrp, setpgrp, setpriority, setquota, setregid, setreuid, setrlimit, setsid, setsockopt, settimeofday, setuid, shmsys, shutdown, sigblock, sigpause, sigpending, sigsetmask, sigstack, sigsys, sigvec, **socket**, socketaddr, socketpair, sstk, stat, stat, statfs, stime, stty, swapon, symlink, sync, sysconf, time, times, truncate, umask, umount, uname, unlink, unmount, ustat, utime, utimes, vadvise, vfork, vhangup, vlimit, vpixsys, vread, vtimes, vtrace, vwrite, wait, wait3, wait4, write, writev

Exemplo: na Rede



- O detetor corre em modo promíscuo numa máquina e analisa todo o tráfego numa rede
- **Dificuldades:**
  - como as máquinas muitas vezes estão ligadas a comutadores (i.e., switches), torna-se difícil escutar todas as ligações a não ser que haja suporte de hardware
  - o conteúdo de tráfego cifrado não pode ser observado

Fiabilidade

- Embora seja muito difícil comparar SDI no que respeita à fiabilidade, este é um conceito que teoricamente é muito importante na determinação do melhor SDI para um determinado ambiente
- Neste aspeto temos de considerar as questões
  - ***falsos positivos*** (falso alarme) : resultam na perca de recursos, uma vez que alguém tem de investigar a potencial intrusão que na realidade é inexistente
  - ***falsos negativos*** (não deteção) : o SDI não deteta uma intrusão (ou demora muito tempo a detetar) o que pode resultar na divulgação de informação secreta, destruição de recursos, ...
- A fiabilidade depende, por exemplo,
  - da capacidade do SDI conseguir ou não processar todos os eventos (suporte de hardware, se o sistema gera muitos eventos, ...)
  - da qualidade da informação associada a cada evento (consigo seguir um ataque que resulta do acesso a vários programas, potencialmente de mais do que um utilizador)
  - do padrão ser ou não conhecido (IDS baseado no conhecimento) **ou** dos utilizadores comportarem-se ou não de uma forma regular (IDS baseado no comportamento)

Acção após Detecção

- Após a deteção e identificação de uma intrusão deve haver uma resposta
- O mecanismo empregue na resposta depende
  - ambiente operacional (existe ou não sempre alguém responsável pela segurança)
  - função/importância do sistema (o servidor com informação médica é ou não usado durante uma operação)

- **Respostas passivas** : informa-se o administrador do ataque e espera-se que ele o corrija (método mais utilizado até à poucos anos)
  - *alarmes e notificações* : gera-se um alarme usando-se uma das variadas formas disponíveis (e.g., email, janela no monitor, um alarme sonoro)
  - *SNMP traps* : se existir uma interligação entre o sistema de gestão de rede e o SDI, pode-se usar o mesmo suporte para divulgar a intrusão

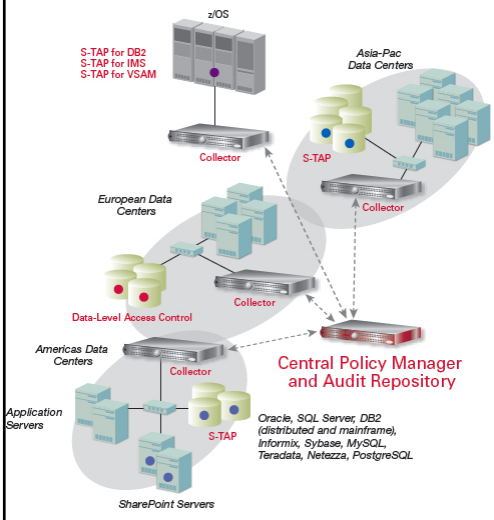
Acção após Detecção

- **Resposta ativa** : o sistema bloqueia ou afeta o progresso do ataque
  - *iniciar ação contra o intruso* : pode tomar diversas formas, mais ou menos benignas
    - » terminar a ligação TCP do intruso
    - » bloquear através do *router* ou *firewall* pacotes provenientes do IP do intruso
    - » fazer uma ataque contra a máquina e rede do intruso

Esta última solução, por mais tentadora que pareça, tem de ser ponderada porque
- *normalmente o ataque é proveniente dum sistema de outra vítima, que em seguida nos pode colocar um processo em tribunal*
- *muitos dos ataques usam IP spoofing*
- *a nossa resposta pode resultar num escalar de ataques pelo adversário*

  - *alterar o ambiente* :
    - » determinar e corrigir as causas que permitiram a intrusão ("*self-healing*")
  - *recolher mais informação* :
    - » alterar o funcionamento do SDI para que ele consiga monitorizar mais cuidadosamente o sistema
    - » recolher mais informação sobre o ataque para que se possa levar o atacante a responder em tribunal (usa-se por exemplo um "*honey pot*")

SIEM: Security Information and Event Monitoring Systems



Bibliografia

- Stallings & Brown, Computer Security: Principles and Practice, Third Edition, 2014
  - Leitura obrigatória: cap 8
  - Leitura opcional: