# Capítulo 8

# Mecanismos de auditoria

## 8.1 Introdução

Na área de segurança de sistemas, o termo "auditar" significa recolher dados sobre o funcionamento de um sistema ou aplicação e analisá-los para descobrir vulnerabilidades ou violações de segurança, ou para examinar violações já constatadas, buscando suas causas e possíveis consequências[1] [Sandhu and Samarati, 1996]. Os dois pontos-chave da auditoria são portanto a *coleta* de dados e a *análise* desses dados, que serão discutidas a seguir.

## 8.2 Coleta de dados

Um sistema computacional em funcionamento processa uma grande quantidade de eventos. Destes, alguns podem ser de importância para a segurança do sistema, como a autenticação de um usuário (ou uma tentativa mal-sucedida de autenticação), uma mudança de credenciais, o lançamento ou encerramento de um serviço, etc. Os dados desses eventos devem ser coletados a partir de suas fontes e registrados de forma adequada para a análise e arquivamento.

Dependendo da natureza do evento, a coleta de seus dados pode ser feita no nível da aplicação, de subsistema ou do núcleo do sistema operacional:

**Aplicação:** eventos internos à aplicação, cuja semântica é específica ao seu contexto. Por exemplo, as ações realizadas por um servidor HTTP, como páginas fornecidas, páginas não encontradas, erros de autenticação, pedidos de operações não suportadas, etc. Normalmente esses eventos são registrados pela própria aplicação, muitas vezes usando formatos próprios para os dados.

**Subsistema:** eventos não específicos a uma aplicação, mas que ocorrem no espaço de usuário do sistema operacional. Exemplos desses eventos são a autenticação de usuários (ou erros de autenticação), lançamento ou encerramento de serviços do sistema, atualizações de softwares ou de bibliotecas, criação ou remoção de usuários, etc. O registro desses eventos normalmente fica a cargo dos processos ou bibliotecas responsáveis pelos respectivos subsistemas.

[1] A análise de violações já ocorridas é comumente conhecida como *análise postmortem*.

**Núcleo:** eventos que ocorrem dentro do núcleo do sistema, sendo inacessíveis aos processos. É o caso dos eventos envolvendo o hardware, como a detecção de erros ou mudança de configurações, e de outros eventos internos do núcleo, como a criação de *sockets* de rede, semáforos, área de memória compartilhada, reinicialização do sistema, etc.

**Rede:** os eventos são oriundos da rede: estabelecimento de conexões TCP, pacotes de gerenciamento ICMP, varredura de hosts e portas, etc. A coleta de eventos pode envolver apenas um segmento de rede ou toda a rede corporativa, fazendo uso de coletores ("sondas") colocadas em posições estratégicas da rede, como roteadores.

Um aspecto importante da coleta de dados para auditoria é sua forma de representação. A abordagem mais antiga e comum, amplamente disseminada, é o uso de arquivos de registro (*logfiles*). Um arquivo de registro contém uma sequência cronológica de descrições textuais de eventos associados a uma fonte de dados, geralmente uma linha por evento. Um exemplo clássico dessa abordagem são os arquivos de registro do sistema UNIX; a listagem a seguir apresenta um trecho do conteúdo do arquivo `/var/log/security`, geralmente usado para reportar eventos associados à autenticação de usuários:

```
...
Sep  8 23:02:09 espec sudo: e89602174 : user NOT in sudoers ; TTY=pts/1 ; USER=root ; COMMAND=/bin/su
Sep  8 23:19:57 espec userhelper[20480:] running '/sbin/halt' with user_u:system_r:hotplug_t context
Sep  8 23:34:14 espec sshd[6302:] pam_unix(sshd:auth): failure; rhost=210.210.102.173  user=root
Sep  8 23:57:16 espec sshd[6302:] Failed password for root from 210.103.210.173 port 14938 ssh2
Sep  8 00:08:16 espec sshd[6303:] Received disconnect from 210.103.210.173: 11: Bye Bye
Sep  8 00:35:24 espec gdm[9447:] pam_unix(gdm:session): session opened for user rodr by (uid=0)
Sep  8 00:42:19 espec gdm[857:] pam_unix(gdm:session): session closed for user rafael3
Sep  8 00:49:06 espec userhelper[11031:] running '/sbin/halt' with user_u:system_r:hotplug_t context
Sep  8 00:53:40 espec gdm[12199:] pam_unix(gdm:session): session opened for user rafael3 by (uid=0)
Sep  8 00:53:55 espec gdm[12199:] pam_unix(gdm:session): session closed for user rafael3
Sep  8 01:08:43 espec gdm[9447:] pam_unix(gdm:session): session closed for user rodr
Sep  8 01:12:41 espec sshd[14125:] Accepted password for rodr from 189.30.227.212 port 1061 ssh2
Sep  8 01:12:41 espec sshd[14125:] pam_unix(sshd:session): session opened for user rodr by (uid=0)
Sep  8 01:12:41 espec sshd[14127:] subsystem request for sftp
Sep  8 01:38:26 espec sshd[14125:] pam_unix(sshd:session): session closed for user rodr
Sep  8 02:18:29 espec sshd[17048:] Accepted password for e89062004 from 20.0.0.56 port 54233 ssh2
Sep  8 02:18:29 espec sshd[17048:] pam_unix(sshd:session): session opened for user e89062004 by (uid=0)
Sep  8 02:18:29 espec sshd[17048:] pam_unix(sshd:session): session closed for user e89062004
Sep  8 09:06:33 espec sshd[25002:] Postponed publickey for mzr from 159.71.224.62 port 52372 ssh2
Sep  8 06:06:34 espec sshd[25001:] Accepted publickey for mzr from 159.71.224.62 port 52372 ssh2
Sep  8 06:06:34 espec sshd[25001:] pam_unix(sshd:session): session opened for user mzr by (uid=0)
Sep  8 06:06:57 espec su: pam_unix(su-l:session): session opened for user root by mzr(uid=500)
...
```

A infraestrutura tradicional de registro de eventos dos sistemas UNIX é constituída por um *daemon*[2] chamado `syslogd` (*System Log Daemon*). Esse *daemon*[3] usa um *socket* local e um socket UDP para receber mensagens descrevendo eventos, geradas pelos demais subsistemas e aplicações através de uma biblioteca específica. Os eventos são descritos por mensagens de texto e são rotulados por suas fontes em *serviços* (`AUTH`, `KERN`, `MAIL`, etc.) e *níveis* (`INFO`, `WARNING`, `ALERT`, etc.). A partir de seu arquivo de

[2]Processo que executa em segundo plano, sem estar associado a uma interface com o usuário, como um terminal ou janela.

[3]*Daemons* são processos que executam continuamente, sem interação com o usuário, para prover serviços ao sistema operacional, como gestão de impressoras, de conexões de rede, etc. A palavra *daemon* vem do grego antigo e significa "ser sobrenatural" ou "espírito".

configuração, o processo `syslogd` registra a data de cada evento recebido e decide seu destino: armazenar em um arquivo, enviar a um terminal, avisar o administrador, ativar um programa externo ou enviar o evento a um *daemon* em outro computador são as principais possibilidades. A Figura 8.1 apresenta os principais componentes dessa arquitetura.

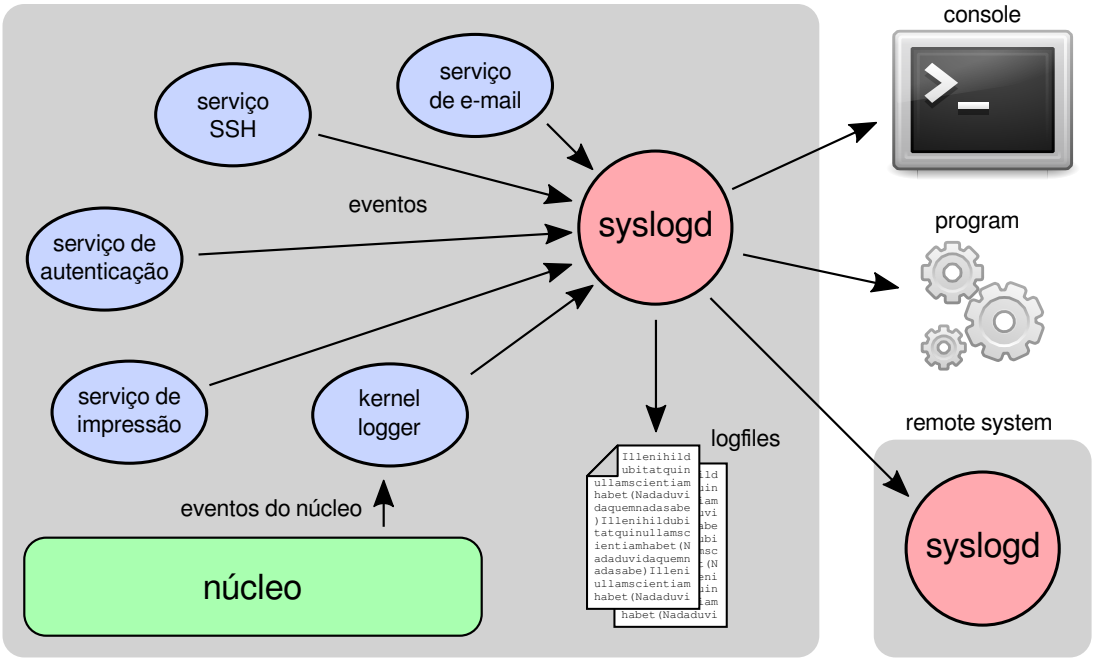


Figura 8.1: O serviço de *logs* em UNIX.

Os sistemas Windows mais recentes usam uma arquitetura similar, embora mais sofisticada do ponto de vista do formato dos dados, pois os eventos são descritos em formato XML (a partir do Windows Vista). O serviço *Windows Event Log* assume o papel de centralizador de eventos, recebendo mensagens de várias fontes, entre elas os componentes do subsistema de segurança (LSASS e SRM, Seção 7.3), as aplicações e o próprio núcleo. Conforme visto anteriormente, o componente LSASS gera eventos relativos à autenticação dos usuários, enquanto o SRM registra os acessos a cada objeto de acordo com as regras de auditoria definidas em sua SACL (*System ACLs*). Além disso, aplicações externas podem se registrar junto ao sistema de logs para receber eventos de interesse, através de uma interface de acesso baseada no modelo *publish/subscribe*.

Além dos exemplos aqui apresentados, muitos sistemas operacionais implementam arquiteturas específicas para auditoria, como é o caso do BSM (*Basic Security Module*) do sistema Solaris e sua implementação OpenBSM para o sistema operacional OpenBSD. O sistema MacOS X também provê uma infraestrutura de auditoria, na qual o administrador pode registrar os eventos de seu interesse e habilitar a geração de registros.

Além da coleta de eventos do sistema à medida em que eles ocorrem, outras formas de coleta de dados para auditoria são frequentes. Por exemplo, ferramentas de segurança podem vasculhar o sistema de arquivos em busca de arquivos com conteúdo malicioso, ou varrer as portas de rede para procurar serviços suspeitos.

## 8.3 Análise de dados

Uma vez registrada a ocorrência de um evento de interesse para a segurança do sistema, deve-se proceder à sua análise. O objetivo dessa análise é sobretudo identificar possíveis violações da segurança em andamento ou já ocorridas. Essa análise pode ser feita sobre os registros dos eventos à medida em que são gerados (chamada análise *online*) ou sobre registros previamente armazenados (análise *offline*). A análise *online* visa detectar problemas de segurança com rapidez, para evitar que comprometam o sistema. Como essa análise deve ser feita simultaneamente ao funcionamento do sistema, é importante que seja rápida e leve, para não prejudicar o desempenho do sistema nem interferir nas operações em andamento. Um exemplo típico de análise online são os antivírus, que analisam os arquivos à medida em que estes são acessados pelos usuários.

Por sua vez, a análise *offline* é realizada com dados previamente coletados, possivelmente de vários sistemas. Como não tem compromisso com uma resposta imediata, pode ser mais profunda e detalhada, permitindo o uso de técnicas de mineração de dados para buscar correlações entre os registros, que possam levar à descoberta de problemas de segurança mais sutis. A análise offline é usada em sistemas de detecção de intrusão, por exemplo, para analisar a história do comportamento de cada usuário. Além disso, é frequentemente usada em sistemas de informação bancários, para se analisar o padrão de uso dos cartões de débito e crédito dos correntistas e identificar fraudes.

As ferramentas de análise de registros de segurança podem adotar basicamente duas abordagens: análise por assinaturas ou análise por anomalias. Na *análise por assinaturas*, a ferramenta tem acesso a uma base de dados contendo informações sobre os problemas de segurança conhecidos que deve procurar. Se algum evento ou registro se encaixar nos padrões descritos nessa base, ele é considerado uma violação de segurança. Um exemplo clássico dessa abordagem são os programas antivírus: um antivírus típico varre o sistema de arquivos em busca de conteúdos maliciosos. O conteúdo de cada arquivo é verificado junto a uma *base de assinaturas*, que contém descrições detalhadas dos vírus conhecidos pelo software; se o conteúdo de um arquivo coincidir com uma assinatura da base, aquele arquivo é considerado suspeito. Um problema com essa forma de análise é sua incapacidade de detectar novas ameaças, como vírus desconhecidos, cuja assinatura não esteja na base.

Por outro lado, uma ferramenta de *análise por anomalias* conta com uma base de dados descrevendo o que se espera como comportamento ou conteúdo normal do sistema. Eventos ou registros que não se encaixarem nesses padrões de normalidade são considerados como violações potenciais da segurança, sendo reportados ao administrador do sistema. A análise por anomalias, também chamada de análise baseada em heurísticas, é utilizada em certos tipos de antivírus e sistemas de detecção de intrusão, para detectar vírus ou ataques ainda desconhecidos. Também é muito usada em sistemas de informação bancários, para detectar fraudes envolvendo o uso das contas e cartões bancários. O maior problema com esta técnica é caracterizar corretamente o que se espera como comportamento "normal", o que pode ocasionar muitos erros.

## 8.4 Auditoria preventiva

Além da coleta e análise de dados sobre o funcionamento do sistema, a auditoria pode agir de forma "preventiva", buscando problemas potenciais que possam comprometer a segurança do sistema. Há um grande número de ferramentas de auditoria, que abordam aspectos diversos da segurança do sistema, entre elas [Pfleeger and Pfleeger, 2006]:

- *Vulnerability scanner*: verifica os softwares instalados no sistema e confronta suas versões com uma base de dados de vulnerabilidades conhecidas, para identificar possíveis fragilidades. Pode também investigar as principais configurações do sistema, com o mesmo objetivo. Como ferramentas deste tipo podem ser citadas: *Metasploit*, *Nessus Security Scanner* e SAINT (*System Administrator's Integrated Network Tool*).

- *Port scanner*: analisa as portas de rede abertas em um computador remoto, buscando identificar os serviços de rede oferecidos pela máquina, as versões do softwares que atendem esses serviços e a identificação do próprio sistema operacional subjacente. O *NMap* é provavelmente o *scanner* de portas mais conhecido atualmente.

- *Password cracker*: conforme visto na Seção 5.4, as senhas dos usuários de um sistema são armazenadas na forma de resumos criptográficos, para aumentar sua segurança. Um "quebrador de senhas" tem por finalidade tentar descobrir as senhas dos usuários, para avaliar sua robustez. A técnica normalmente usada por estas ferramentas é o ataque do dicionário, que consiste em testar um grande número de palavras conhecidas, suas variantes e combinações, confrontando seus resumos com os resumos das senhas armazenadas. Quebradores de senhas bem conhecidos são o *John the Ripper* para UNIX e *Cain and Abel* para ambientes Windows.

- *Rootkit scanner*: visa detectar a presença de *rootkits* (vide Seção 1.2) em um sistema, normalmente usando uma técnica *offline* baseada em assinaturas. Como os *rootkits* podem comprometer até o núcleo do sistema operacional instalado no computador, normalmente as ferramentas de detecção devem ser aplicadas a partir de outro sistema, carregado a partir de uma mídia externa confiável (CD ou DVD).

- *Verificador de integridade*: a segurança do sistema operacional depende da integridade do núcleo e dos utilitários necessários à administração do sistema. Os verificadores de integridade são programas que analisam periodicamente os principais arquivos do sistema operacional, comparando seu conteúdo com informações previamente coletadas. Para agilizar a verificação de integridade são utilizadas somas de verificação (*checksums*) ou resumos criptográficos como o MD5 e SHA1. Essa verificação de integridade pode se estender a outros objetos do sistema, como a tabela de chamadas de sistema, as portas de rede abertas, os processos de sistema em execução, o cadastro de softwares instalados, etc. Um exemplo clássico de ferramenta de verificação de integridade é o *Tripwire* [Tripwire, 2003], mas existem diversas outras ferramentas mais recentes com propósitos similares.

# Exercícios

1. Sistemas de detecção de intrusão (IDS - *Intrusion Detection Systems*) podem ser classificados sob várias perspectivas. Explique como os IDSs são classificados segundo:

   (a) A origem dos dados analisados;

   (b) O momento da análise dos dados;

   (c) A forma de análise dos dados.

# Referências

C. Pfleeger and S. L. Pfleeger. *Security in Computing, 4th Edition*. Prentice Hall PTR, 2006.

R. Sandhu and P. Samarati. Authentication, access control, and audit. *ACM Computing Surveys*, 28(1), 1996.

Tripwire. The Tripwire open source project. http://www.tripwire.org, 2003.