

Universidade do Minho
Escola de Engenharia

Manuel Alexandre Pinho Rocha

Modelo para definição de criticidade em eventos de segurança em redes de computadores.



Universidade do Minho
Escola de Engenharia

Manuel Alexandre Pinho Rocha

Modelo para definição de criticidade em
eventos de segurança em redes de
computadores.

Tese de Mestrado
Ciclo de Estudos Integrados Conducentes ao
Grau de Mestre em Engenharia de Comunicações

Trabalho efetuado sob a orientação do
Professor Doutor Henrique Manuel Dinis Santos

DECLARAÇÃO

Nome: Manuel Alexandre Pinho Rocha

Correio electrónico: a51069@alunos.uminho.pt

Tel./Tlm.: 939063013

Número do Bilhete de Identidade: 13394628

Título da dissertação: Modelo para definição de criticidade em eventos de segurança em redes de computadores.

Ano de conclusão: 2013

Orientador: Professor Doutor Henrique Manuel Dinis Santos

Designação do Mestrado:

Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia de Comunicações

Área de Especialização: Engenharia de Comunicações

Escola: Escola de Engenharia

Departamento: Sistemas de Informação/ALGORITMI

1. É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA DISSERTAÇÃO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.
2. É AUTORIZADA A REPRODUÇÃO PARCIAL DESTA DISSERTAÇÃO (indicar, caso tal seja necessário, nº máximo de páginas, ilustrações, gráficos, etc.), APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.
3. De acordo com a legislação em vigor, não é permitida a reprodução de qualquer parte desta dissertação

Guimarães, 17/12/2013

Assinatura: _____

Modelo para definição de criticidade em
eventos de segurança de redes de
computadores

Manuel Rocha

14 de Dezembro de 2013

Agradecimentos

A realização deste trabalho só foi possível devido à intervenção, directa ou indirecta, de algumas pessoas, a quem gostaria de aqui mencionar e agradecer:

Ao meu orientador, professor Henrique Santos, pela disponibilidade e pelo entusiasmo transmitido para que este trabalho chegá-se a bom porto.

Aos meus pais e à minha irmã por todo o apoio, carinho e paciência ao longo de toda a minha vida e por me ajudarem a tornar na pessoa que sou hoje.

Aos meus amigos César Oliveira, Hélder Ribeiro, Hugo Leite, João Brito, João Pinheiro, Luís Nascimento, Mário Dias, Ricardo Maciel, Rui Rodrigues, Sérgio Oliveira, Tiago Pimenta, entre outros pela amizade e por todos os bons momentos.

Por fim gostaria de agradecer à Sara por todo o apoio e carinho, mas sobretudo por todo o amor que me faz sentir completo.

Resumo

A invasão de redes de computador é hoje um dos maiores problemas nas várias organizações que zelam pela segurança da sua informação, bem como pela operacionalidade dos seus sistemas. As várias ferramentas disponíveis que visam colmatar e atenuar este problema, nem sempre são suficientes ou não traduzem de forma clara a informação relativa às várias invasões a que uma rede ou um sistema está exposto.

As ferramentas existentes revelam um défice em relação à informação dos eventos de segurança. A informação que derivada dos respectivos eventos é pouco específica e pouco perceptiva em relação ao impacto dos mesmos, criando uma barreira muito grande na gestão e administração de sistemas e redes de computadores ao nível da detecção de intrusões.

Modelo para definição de criticidade em eventos de segurança de redes de computadores é o tema em que se enquadra este projecto, que propõe um método de classificação de eventos de segurança. Neste projecto pretende-se criar um método capaz de classificar eventos de segurança, uma classificação com base na refinação de informação de contexto, de forma a obter a criticidade e perigosidade de eventos de segurança.

PALAVRAS-CHAVE: Detecção de intrusões, Segurança da informação, Segurança em redes de computadores, gestão de eventos de segurança

Abstract

Intrusions in computer networks are a major problem for the organizations that care about the security of the information and the operability of their systems. Concerning the innumerable threats that a network or a system can be exposed, several tools are available on the market to address and to mitigate these issues.

However, those tools either aren't sufficient or aren't capable to translate the information on a clear way because of the deficit for generating sufficient information when security events are triggered. Usually, the information retrieved from these events is not very specific and is difficult for security analysts to comprehend which can be a major obstacle for the management and administration of computer networks, regarding the detection of intrusions.

The management of security events on information systems is under the main objective of this dissertation and a model to classify these events is proposed, implemented and tested. The classification is based on the refinement of context information in order to obtain the criticism and the level of dangerousness on security events.

KEY WORDS: Intrusion detection, Information Security, Computer network security, Security events management.

Lista de acronimos

| | |
|--------------|---|
| AIDS | <i>Anomaly Intrusion Detection System</i> |
| BGP | <i>Exterior Gateway Protocol</i> |
| CAPEC | <i>Common Attack Pattern Enumeration and Classification</i> |
| DNS | <i>Domain Name System</i> |
| DoS | <i>Denial of Service</i> |
| FTP | <i>File Transfer Protocol</i> |
| HIDS | <i>Host Intrusion Detection System</i> |
| HTTP | <i>Hypertext Transfer Protocol</i> |
| ICMP | <i>Internet Control Message Protocol</i> |
| IDS | <i>Intrusion Detection System</i> |
| IP | <i>Internet Protocol</i> |
| NIDS | <i>Network Intrusion Detection System</i> |
| OSSIM | <i>Open Source Security Information Management</i> |
| OSPF | <i>Open Shortest Path First</i> |
| RIP | <i>Routing Information Protocol</i> |
| SEM | <i>Security Event Management</i> |
| SIDS | <i>Signature Intrusion Detection System</i> |

SIEM *Security Information and Event Management*

SIM *Security Information Management*

SMTP *Simple Mail Transfer Protocol*

SNMP *Simple Network Management Protocol*

TCP *Transmission Control Protocol*

TCP/IP *Transmission Control Protocol/Internet Protocol*

UDP *User Datagram Protocol*

XML *eXtensible Markup Language*

Conteúdo

| | |
|--|-------------|
| Conteúdo | xi |
| Lista de Figuras | xv |
| Lista de Tabelas | xvii |
| 1 Introdução | 1 |
| 1.1 Enquadramento e motivação | 1 |
| 1.2 Objectivos | 4 |
| 1.3 Métodos de investigação | 4 |
| 1.4 Estrutura do documento | 7 |
| 2 Fundamentos teóricos de segurança de redes de computadores | 9 |
| 2.1 Fundamentos das redes de comunicações IP | 9 |
| 2.2 Tráfego TCP/IP e as suas vulnerabilidades | 11 |
| 2.3 Segurança em Redes de Computadores | 13 |
| 2.3.1 Detecção de intrusões | 14 |
| 2.3.1.1 Registo de eventos de segurança | 17 |
| 2.3.2 Modelação de ataques | 18 |
| 2.3.2.1 <i>Attack trees</i> | 19 |
| 2.3.2.2 <i>Attack graphs</i> | 20 |
| 2.3.2.3 Common Attack Pattern Enumeration and Classification | 23 |
| 2.3.3 Modelação de vulnerabilidades | 25 |
| 2.3.4 <i>Security Information and Event Management</i> | 25 |
| 2.4 Ferramentas/Frameworks | 27 |
| 2.4.1 <i>Snort</i> | 28 |

| | | |
|----------|--|-----------|
| 2.4.2 | <i>Ossec</i> | 29 |
| 2.4.3 | <i>Open Source Security Information Management</i> | 31 |
| 2.4.4 | <i>Bro</i> | 32 |
| 3 | Especificação e implementação do sistema | 35 |
| 3.1 | Requisitos do sistema | 35 |
| 3.1.1 | Cenários e contextos | 36 |
| 3.1.2 | Requisitos funcionais | 36 |
| 3.1.3 | Requisitos não funcionais | 37 |
| 3.2 | Modelo de classificação e valorização de eventos | 38 |
| 3.2.1 | Vulnerabilidades | 38 |
| 3.2.1.1 | Associação de vulnerabilidades | 39 |
| 3.2.2 | Ataques | 40 |
| 3.2.3 | Criticidade | 40 |
| 3.2.3.1 | Perigosidade | 40 |
| 3.2.3.2 | Tipos de ataque | 42 |
| 3.2.3.3 | Reputação IP | 43 |
| 3.3 | Arquitectura e implementação do sistema | 43 |
| 3.3.1 | Arquitectura do sistema | 44 |
| 3.3.1.1 | Módulo recolha de eventos | 44 |
| 3.3.1.2 | Módulo reputação IP | 44 |
| 3.3.1.3 | Módulo vulnerabilidades | 45 |
| 3.3.1.4 | Módulo ataques | 45 |
| 3.3.1.5 | Módulo valorização e classificação | 45 |
| 3.3.2 | Implementação do sistema | 45 |
| 3.3.2.1 | Classes da aplicação | 46 |
| 3.3.2.2 | Algoritmos do modelo desenvolvido | 47 |
| 4 | Testes e avaliações | 53 |
| 4.1 | Ambiente de teste | 53 |
| 4.2 | Testes e Resultados | 55 |
| 4.2.1 | Tipos de ataques | 56 |
| 4.2.2 | Resultados obtidos | 58 |
| 4.2.2.1 | Teste 1 - Resultados obtidos | 58 |
| 4.2.2.2 | Teste 2 - Resultados obtidos | 61 |
| 4.2.2.3 | Teste 3 - Resultados obtidos | 62 |
| 5 | Conclusão e trabalho futuro | 65 |
| 5.1 | Conclusões do projecto | 65 |
| 5.2 | Análise crítica | 66 |
| 5.3 | Trabalho futuro | 67 |

| | |
|---|------------|
| Bibliografia | 69 |
| Apêndice A Classes da aplicação | 75 |
| Apêndice B Imagens da aplicação | 79 |
| Apêndice C Tabelas de resultados | 83 |
| Apêndice D Tabelas de resultados | 93 |
| Apêndice E Tabelas de resultados | 101 |

Lista de Figuras

| | | |
|------|---|----|
| 1.1 | Vulnerabilidades catalogadas | 2 |
| 1.2 | Ataques mais frequentes | 3 |
| 1.3 | Metodologia de investigação <i>Design Science</i> | 6 |
| 2.1 | Pilha protocolar TCP/IP | 11 |
| 2.2 | Mapa da <i>Internet</i> | 12 |
| 2.3 | Etapas de um ataque | 18 |
| 2.4 | Exemplo de <i>Attack Tree</i> | 20 |
| 2.5 | Exemplo de um <i>Attack Graph: Host-based Network Attack Graph</i> | 23 |
| 2.6 | Arquitectura conceptual dos SIEM | 26 |
| 2.7 | Arquitectura genérica dos SIEM | 28 |
| 2.8 | Arquitectura do <i>Snort</i> | 29 |
| 2.9 | Arquitectura do <i>Ossec</i> | 30 |
| 2.10 | Arquitectura do <i>OSSIM</i> | 32 |
| 2.11 | Arquitectura do <i>Bro</i> | 33 |
| 3.1 | Arquitectura do sistema implementado | 44 |
| 3.2 | Diagrama de classes | 46 |
| A.1 | Classe <i>DataBase</i> | 75 |
| A.2 | Classe <i>XmlParserCVE</i> | 76 |
| A.3 | Classe <i>XmlParserCapec</i> | 77 |
| A.4 | Classe <i>XmlParserNessus</i> | 78 |
| A.5 | Classe <i>ReputationIP</i> | 78 |
| A.6 | Classe <i>GUI</i> | 78 |
| B.1 | Interface gráfica - preenchimento dos parâmetros relativos aos tipos de ataque(50%) | 79 |

| | | |
|-----|---|----|
| B.2 | Interface gráfica - preenchimento dos parâmetros relativos aos tipos de ataque(95%) | 80 |
| B.3 | Interface gráfica - preenchimento dos parâmetros relativos aos tipos de ataque(20%) | 81 |
| B.4 | Interface gráfica da ferramenta <i>Armitage</i> | 81 |

Lista de Tabelas

| | | |
|------|---|----|
| 1.1 | Número de documentos relativos ao <i>survey</i> bibliográfico | 5 |
| 1.2 | Elementos bibliográficos mais relevantes | 6 |
| 2.1 | Classificação dos IDS | 17 |
| 4.1 | Serviços instalados nos diferentes <i>hosts</i> | 55 |
| 4.2 | Testes e resultados - <i>scanning</i> | 59 |
| C.1 | Ataques realizados em cada <i>host</i> | 84 |
| C.2 | Testes e resultados - <i>Windows XP Sp2</i> | 85 |
| C.3 | Testes e resultados - <i>Windows XP Sp2</i> | 85 |
| C.4 | Testes e resultados - <i>Windows XP Sp2</i> | 86 |
| C.5 | Testes e resultados - <i>Windows XP Sp2</i> | 86 |
| C.6 | Testes e resultados - <i>Windows XP Sp2</i> | 86 |
| C.7 | Testes e resultados - <i>Windows XP Sp2</i> | 87 |
| C.8 | Testes e resultados - <i>Windows Server 2008</i> | 87 |
| C.9 | Testes e resultados - <i>Windows Server 2008</i> | 87 |
| C.10 | Testes e resultados - <i>Windows Server 2008</i> | 88 |
| C.11 | Testes e resultados - <i>Windows Server 2008</i> | 88 |
| C.12 | Testes e resultados - <i>Windows Server 2008</i> | 88 |
| C.13 | Testes e resultados - <i>Ubuntu 13.04</i> | 89 |
| C.14 | Testes e resultados - <i>Ubuntu 13.04</i> | 89 |
| C.15 | Testes e resultados - <i>Ubuntu 13.04</i> | 89 |
| C.16 | Testes e resultados - <i>Ubuntu 13.04</i> | 90 |
| C.17 | Testes e resultados - <i>Ubuntu 13.04</i> | 90 |
| C.18 | Testes e resultados - <i>CentOS 6</i> | 90 |
| C.19 | Testes e resultados - <i>CentOS 6</i> | 91 |
| D.1 | Testes e resultados - <i>scanning</i> | 94 |

| | | |
|------|--|-----|
| D.2 | Testes e resultados - <i>Windows XP Sp2</i> | 94 |
| D.3 | Testes e resultados - <i>Windows XP Sp2</i> | 95 |
| D.4 | Testes e resultados - <i>Windows XP Sp2</i> | 95 |
| D.5 | Testes e resultados - <i>Windows XP Sp2</i> | 96 |
| D.6 | Testes e resultados - <i>Windows XP Sp2</i> | 96 |
| D.7 | Testes e resultados - <i>Windows XP Sp2</i> | 96 |
| D.8 | Testes e resultados - <i>Windows Server 2008</i> | 96 |
| D.9 | Testes e resultados - <i>Windows Server 2008</i> | 97 |
| D.10 | Testes e resultados - <i>Windows Server 2008</i> | 97 |
| D.11 | Testes e resultados - <i>Windows Server 2008</i> | 98 |
| D.12 | Testes e resultados - <i>Windows Server 2008</i> | 98 |
| D.13 | Testes e resultados - <i>Ubuntu 13.04</i> | 98 |
| D.14 | Testes e resultados - <i>Ubuntu 13.04</i> | 98 |
| D.15 | Testes e resultados - <i>Ubuntu 13.04</i> | 98 |
| D.16 | Testes e resultados - <i>Ubuntu 13.04</i> | 99 |
| D.17 | Testes e resultados - <i>Ubuntu 13.04</i> | 99 |
| D.18 | Testes e resultados - <i>CentOS 6</i> | 99 |
| D.19 | Testes e resultados - <i>CentOS 6</i> | 100 |
| | | |
| E.1 | Testes e resultados - <i>scanning</i> | 102 |
| E.2 | Testes e resultados - <i>Windows XP Sp2</i> | 102 |
| E.3 | Testes e resultados - <i>Windows XP Sp2</i> | 103 |
| E.4 | Testes e resultados - <i>Windows XP Sp2</i> | 103 |
| E.5 | Testes e resultados - <i>Windows XP Sp2</i> | 104 |
| E.6 | Testes e resultados - <i>Windows XP Sp2</i> | 104 |
| E.7 | Testes e resultados - <i>Windows XP Sp2</i> | 104 |
| E.8 | Testes e resultados - <i>Windows Server 2008</i> | 104 |
| E.9 | Testes e resultados - <i>Windows Server 2008</i> | 105 |
| E.10 | Testes e resultados - <i>Windows Server 2008</i> | 105 |
| E.11 | Testes e resultados - <i>Windows Server 2008</i> | 106 |
| E.12 | Testes e resultados - <i>Windows Server 2008</i> | 106 |
| E.13 | Testes e resultados - <i>Ubuntu 13.04</i> | 106 |
| E.14 | Testes e resultados - <i>Ubuntu 13.04</i> | 106 |
| E.15 | Testes e resultados - <i>Ubuntu 13.04</i> | 106 |
| E.16 | Testes e resultados - <i>Ubuntu 13.04</i> | 107 |
| E.17 | Testes e resultados - <i>Ubuntu 13.04</i> | 107 |
| E.18 | Testes e resultados - <i>CentOS 6</i> | 107 |
| E.19 | Testes e resultados - <i>CentOS 6</i> | 108 |

Capítulo 1

Introdução

1.1 Enquadramento e motivação

Todos os dias são usadas novas técnicas de invasão cada vez mais robustas e eficazes, que exploram novas vulnerabilidades dos diversos sistemas computacionais, estando estes constantemente sujeitos a enumeras ameaças que podem dar origem a consequências de diversos tipos. A motivação de quem invade é originada por diversos factores: o simples facto de conseguir algo potencialmente inacessível e inatingível, a obtenção de informação restrita e importante, ou até mesmo a fama, são alguns exemplos do que muitas vezes origina uma invasão.

Na figura 1.1 é possível verificar que o número de vulnerabilidades nos sistemas computacionais aumentou exponencialmente na última década, mantendo-se igualmente distribuído nos últimos anos. Este aumento é preocupante pois o número de ameaças também aumenta, tendo como consequência o crescimento do número de *malware* e ataques cibernéticos.

Segundo o estudo de Richardson (2010/2011), em que participaram 285 organizações, 41.1% dos participantes relatam ter sido vitimas de algum tipo de incidente informático. No mesmo estudo foi perguntado quais os tipos de ataques foram presenciados pelos participantes nas suas organizações, sendo possível verificar na figura 1.2 que os incidentes verificados são de diferentes tipos e em número considerável.

¹Fonte: <http://www.cert.org/stats/> e <http://www.symantec.com/es/es/threatreport/>, visitados em Junho de 2013

²Imagem retirada de Richardson (2010/2011)

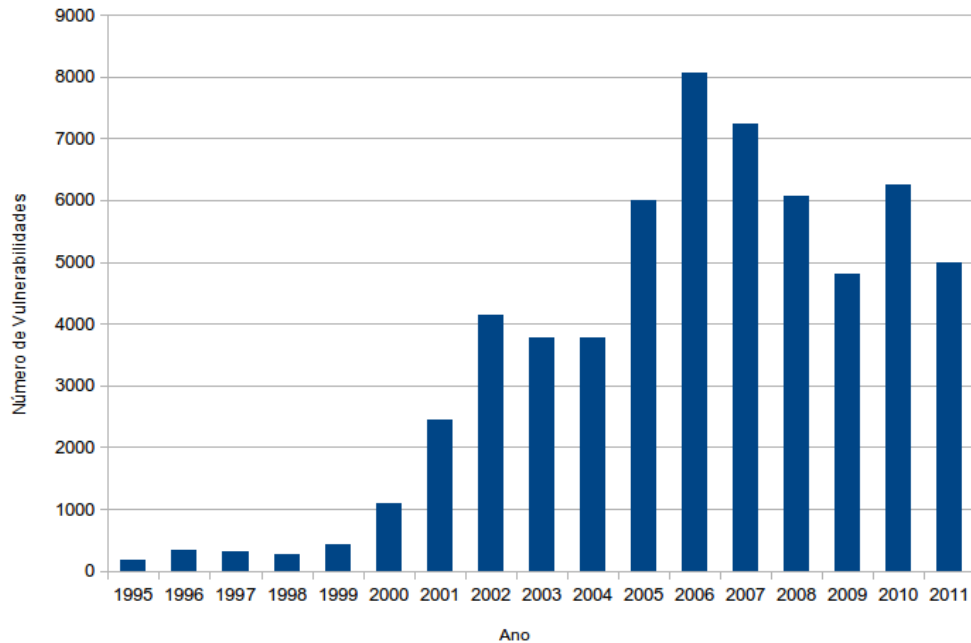
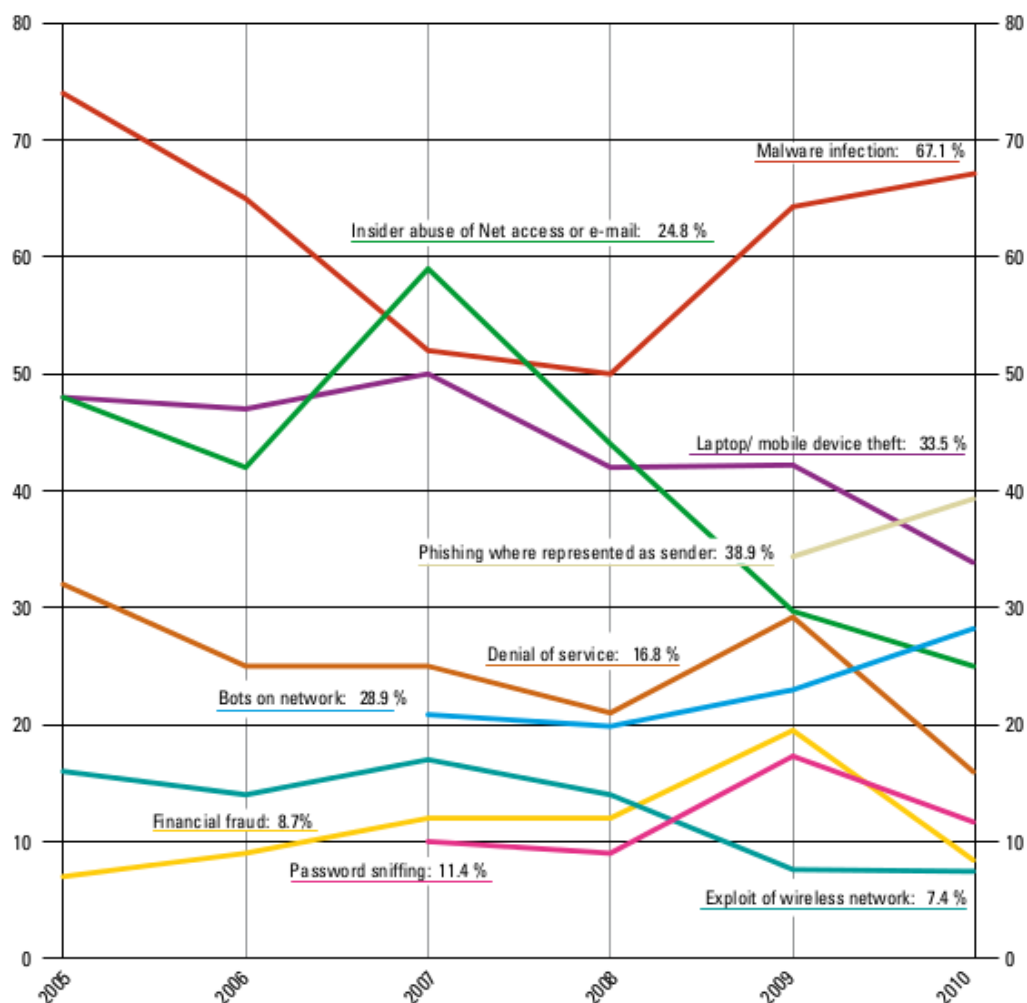


Figura 1.1: Vulnerabilidades catalogadas.¹

Na tentativa de colmatar este problema existem vários tipos de ferramentas, com características diferentes, que apesar de tudo têm as suas limitações. As ferramentas normalmente utilizadas neste tipo de problema geram eventos quando detectado algum comportamento classificado como anormal, de forma a alertar o sucedido. Contudo o número de eventos gerados é normalmente elevado, tornando a tarefa dos gestores e/ou administradores de rede demorada. Para além disso, o facto dos ataques cibernéticos poderem ser de diferentes tipos torna a tarefa de detecção ainda mais complexa.

Os eventos gerados pelas ferramentas de segurança são normalmente registados em *logs*³, de forma a ser possível efectuar auditorias aos sistemas computacionais. Segundo Kent em Souppaya (2006) o número e a variedade de *logs* de segurança tem aumentado consideravelmente devido ao fornecimento contínuo de dados para análise, ao número elevado de fontes de registo e à geração de *logs* redundantes, o que resulta numa auditoria menos eficiente. Para além deste problema, a análise da informação contida nos vários eventos nem sempre é uma tarefa fácil, muitas vezes devido

³Registo de eventos num sistema computacional.

Figura 1.2: Ataques mais frequentes²

à complexidade do conteúdo dessa informação, bem como a forma como a informação é apresentada e descrita. Isto faz com que o gestor e/ou administrador de rede não actue de uma forma rápida e eficiente em caso de anomalia no sistema, expondo assim a informação ao “mundo alheio”, bem como colocando em risco a operacionalidade dos vários sistemas.

Surge então a necessidade de criar ferramentas e mecanismos capazes de analisar e interpretar a informação contida nos diversos eventos, cruzando-a com informação disponível de diversas fontes, de forma a priorizar os eventos em função da sua criticidade. Com isto obtém-se informações mais precisas e completas em relação, por exemplo, a uma possível intrusão ou anomalia

no sistema.

1.2 Objectivos

O objectivo deste projecto foca-se na classificação de eventos de segurança associados a redes de computadores, de forma a criar um modelo de classificação que permita valorizar os diferentes eventos em função do seu tipo. Dado o objectivo principal deste projecto surge naturalmente a seguinte questão de investigação subjacente:

- Como valorizar os eventos de segurança.

Sobre esta questão resultam as seguintes subquestões:

- Que eventos podem e devem ser valorizados
- Quais as fontes de informação utilizadas para a valorização
- Qual a relação entre as diferentes fontes
- Como demonstrar a eficácia da valorização

1.3 Métodos de investigação

Atendendo à natureza do problema apresentado e aos objectivos inicialmente estipulados, recorreu-se ao método *Design Science* (Peffer *et al.* 2007) complementado com *survey* bibliográfico

Para a obtenção da bibliografia, foi seguida uma estratégia de pesquisa com as seguintes *Keywords*: *intrusion detection*, *alert aggregation*, *alert correlation*, *information security*, *event classification*, *log analysis* e *Network security*. Obteve-se assim um número considerável de elementos bibliográficos, relacionada com a grande área do tema deste projecto: Modelo para definição de criticidade em eventos de segurança de redes de computadores. Foram obtidas 58 referências bibliográficas, sendo efectuado uma leitura das mesmas com o objectivo de seleccionar aquelas que abordam as questões deste projecto.

Cada referência bibliográfica com informação relevante, em função da etapa em que o projecto se encontrava, seguia-se uma leitura e análise integrada dessa referência bibliográfica. No final deste procedimento foi efectuada uma

síntese de todas as referências bibliográficas sendo seleccionadas 41 das 58 inicialmente obtidas.

Em relação aos documentos do *survey* bibliográfico, na tabela 1.1 é possível verificar a quantidade por intervalos de ano de publicação.

| Ano do documento | Número de documentos |
|----------------------------|-----------------------------|
| Anterior a 2000 | 3 |
| Entre 2000 e 2005 | 4 |
| Entre 2006 e 2010 | 27 |
| Depois de 2010 | 7 |
| Total de documentos | 41 |

Tabela 1.1: Número de documentos relativos ao *survey* bibliográfico

Contudo nem todos os elementos bibliográficos seleccionados tiveram o mesmo grau de importância no processo de investigação deste projecto. Na tabela 1.2 é possível identificar os temas mais relevantes no âmbito da investigação deste projecto e o número de elementos bibliográfico de cada tema.

Com todos os elementos bibliográficos seleccionados, foi então possível uma melhor compreensão do problema aqui apresentado, de forma a que a abordagem da solução do mesmo fosse a melhor.

Em relação ao método *Design Science*, na figura 1.3 é apresentado todo o processo do método. Tal como descrito em Peffers *et al.* (2007) a primeira fase do processo é identificar o problema que o investigador pretende solucionar e quais as suas principais motivações, demonstrando qual a importância de encontrar uma solução para esse problema. Depois de identificado o problema é necessário definir quais os objectivos para encontrar a sua solução. De acordo com objectivos referentes à solução proposta, o artefacto é desenvolvido e implementado. Por último a avaliação do artefacto desenvolvido em função dos resultados obtidos e da eficiência da solução. No

Modelo para definição de criticidade em eventos de segurança de redes de computadores

| Tema abordado | Número de documentos |
|---|----------------------|
| Detecção de intrusões | 5 |
| Gestão e análise de eventos de segurança | 4 |
| Vulnerabilidades em sistemas computacionais | 3 |
| Modelação de ataques cibernéticos | 6 |
| Total de documentos | 18 |

Tabela 1.2: Elementos bibliográficos mais relevantes

final, de forma a terminar todo o projecto, a respectiva publicação de modo a transmitir todo o conhecimento adquirido.

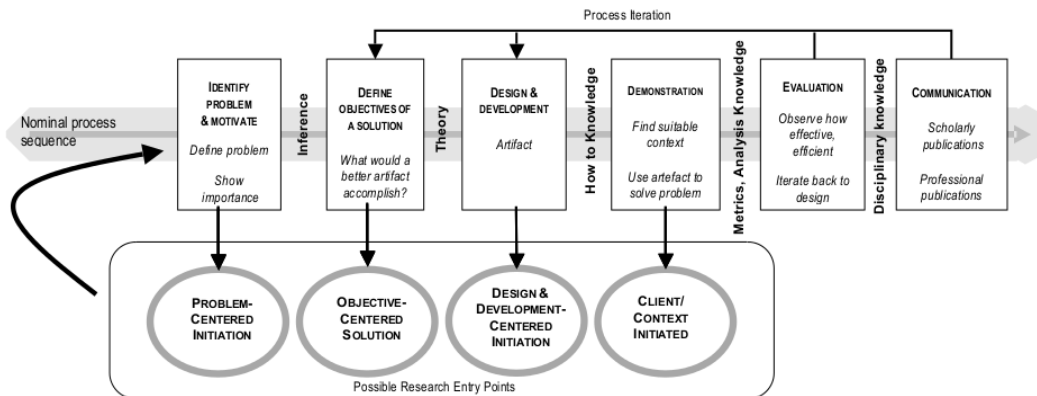


Figura 1.3: Metodologia de investigação *Design Science*: Imagem retirada de Peffers *et al.* (2007)

Para validação do trabalho proposto, bem como de todas as conclusões dele derivadas e demonstrar a eficiência da solução proposta, recorreu-se a ensaios laboratoriais. Todos os resultados obtidos neste processo são devidamente analisados, documentados e comparados entre si, nos diversos cenários em que este trabalho se enquadra.

O ensaio laboratorial foi efectuado com recurso de uma rede virtual, composta por cinco máquinas virtuais, de forma a criar uma "bancada" de teste a todo o trabalho proposto.

1.4 Estrutura do documento

Nesta secção é apresentada uma descrição de como o presente documento se encontra organizado, explicando resumidamente os capítulos que o compõem.

No primeiro e presente capítulo é efectuado um enquadramento relativo ao tema desta dissertação, e qual a principal motivação para a sua realização. Neste capítulo é ainda especificado o objectivo deste projecto e quais foram os métodos de investigação utilizados para o alcançar.

No capítulo 2 é efectuado um enquadramento de conceitos teóricos relativos ao problema inicialmente identificado com base na revisão de toda a bibliografia seleccionada.

O capítulo 3 consiste essencialmente na especificação e implementação do sistema proposto. Inicialmente são detalhados os requisitos do sistema (funcionais e não funcionais), seguido de uma descrição detalhada do desenho do sistema e das principais componentes que o constituem. Por fim a descrição da arquitectura e implementação do sistema proposto.

No capítulo 4 são apresentados os testes realizados em laboratório ao sistema e as respectivas conclusões, de forma a avaliar a solução proposta.

No capítulo 5 são apresentadas as conclusões de todo o trabalho desenvolvido e também uma análise crítica de todos os aspectos deste trabalho, de forma a salientar os aspectos menos positivos. Por fim é descrito o trabalho futuro do mesmo.

Capítulo 2

Fundamentos teóricos de segurança de redes de computadores

O presente capítulo tem como objectivo enquadrar conceitos teóricos relativos ao tema desta dissertação, nomeadamente segurança em redes de computadores, referenciando também o estado actual de alguns projectos e trabalhos relativos à segurança de redes de computadores, mais concretamente na detecção de intrusões.

Dadas as questões de investigação descritas no capítulo anterior e na tentativa de as responder, é necessário todo um enquadramento teórico em relação à investigação realizada, de forma a conhecer a origem do problema identificado e o que o caracteriza.

2.1 Fundamentos das redes de comunicações IP

As redes de comunicações IP (*Internet Protocol*) são hoje em dia imprescindíveis a qualquer organização. É sobre a rede que flui toda a informação que permite à sociedade e às organizações comunicarem de forma rápida, eficiente e a grandes distâncias. Este tipo de redes assenta sobre um modelo denominado de *Transmission Control Protocol/Internet Protocol* (TCP/IP), que consiste numa pilha protocolar, caracterizada por ser portátil e independente da plataforma usada, onde diversos sistemas podem comunicar entre si, possibilitando assim a ligação entre redes heterogéneas. O protocolo TCP/IP é também conhecido pelo protocolo da *Internet*, conhecida como a "Grande Rede" ou somente "A Rede" (Kurose e Ross 2002).

Como é possível visualizar na figura 2.1, a pilha protocolar TCP/IP é composta por cinco camadas (Kurose en Ross 2002):

- **Camada Física:** É composta pelos componentes físicos que definem o meio de transmissão do canal de comunicação e os respectivos sistemas mecânicos que o compõem, bem como as características dos sinais de comunicação que "passam" pelos mesmos.
- **Camada de Ligação:** Especifica as tecnologias usadas na transferência de dados, bem como os meios de ligação. É responsável pelo controlo de fluxo, especifica a(s) técnica(s) de acesso ao meio, sendo também responsável pela detecção e correcção de erros.
- **Camada de Rede:** É responsável pelo encaminhamento dos datagramas, vulgo pacotes de dados, entre a origem e o destino. Mesmo quando os datagramas tenham que passar por pontos intermédios da rede, a camada de rede calcula a melhor rota para efectuar a entrega dos mesmos. Dos protocolos envolvidos no encaminhamento (interno e externo) destacam-se os seguintes: RIP (*Routing Information Protocol*), OSPF (*Open Shortest Path First*), BGP (*Exterior Gateway Protocol*) e ICMP (*Internet Control Message Protocol*). A camada de rede também é responsável pelo endereçamento, isto é, a atribuição de um endereço único a todas as interfaces dos vários dispositivos da rede.
- **Camada de Transporte:** Permite uma comunicação entre máquinas. É também responsável pelo controlo de fluxo fornecendo um serviço orientado à conexão no caso do protocolo TCP (*Transmission Control Protocol*), e um serviço não orientado à conexão quando utilizado o protocolo UDP (*User Datagram Protocol*). Em relação ao protocolo TCP a entrega de todos os pacotes é efectuada de forma fiável, ou seja, existe uma garantia da entrega de mensagens provenientes da camada superior ao destino. Já no protocolo UDP não há qualquer garantia da entrega dos datagramas.
- **Camada de Aplicação:** É responsável por apoiar as aplicações de rede. A camada de Aplicação especifica os protocolos que directamente servem as aplicações. Os protocolos HTTP, SMTP e FTP são alguns exemplos.

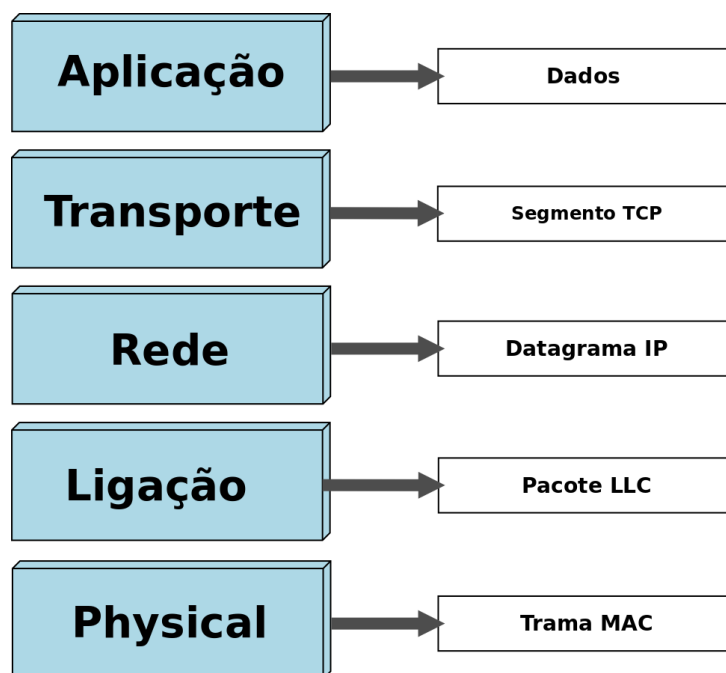


Figura 2.1: Pilha protocolar TCP/IP

2.2 Tráfego TCP/IP e as suas vulnerabilidades

O protocolo TCP/IP foi desenvolvido pelo Departamento de Defesa dos Estados Unidos para a sua rede de pesquisa *ARPANET*, nos meados dos anos 70. Esta rede tinha como principal objectivo conectar centenas de universidades e institutos nos Estados Unidos da América. Contudo, a segurança do protocolo não foi levada em consideração por quem o desenvolveu, muito devido ao ambiente de confiança que se vivia naquela época no departamento onde foi criado. Tal como a segurança dos protocolos, também não foram tomadas medidas de precaução em relação a sua escalabilidade e ao espaço de endereçamento (Li en Jiang 2012).

A *Internet* é hoje um fenómeno de elevada popularidade, universalidade e de escala mundial, sendo possível verificar na figura 2.2 um mapeamento da *Internet*, realizado por um projecto denominado de *The Opte Project*¹. Em função deste fenómeno e de alguns aspectos de segurança outrora descartados, tornaram a *Internet* de alguma forma muito vulnerável a ataques cibernéticos.

¹<http://opte.org/>

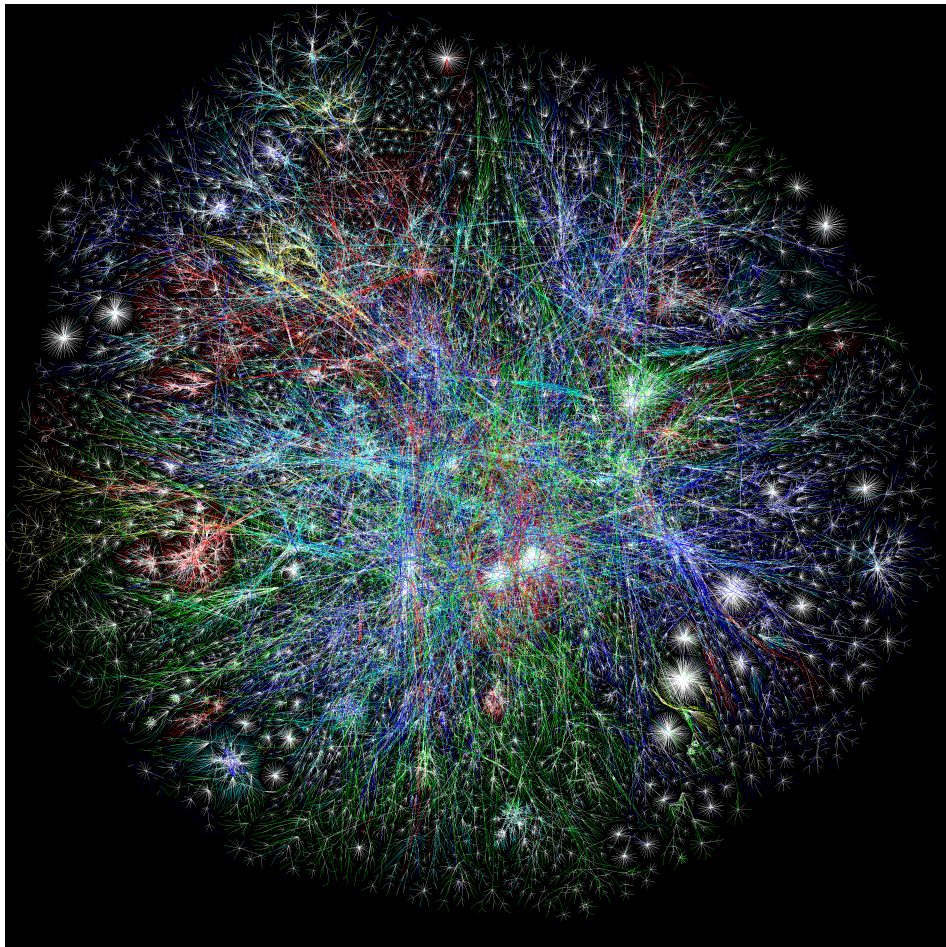


Figura 2.2: Mapa da *Internet* em 2003²

Existe uma variedade de vulnerabilidades na pilha protocolar TCP/IP, que derivam da forma como a pilha protocolar foi "desenhada" e "projectada". Esses problemas de segurança são provenientes dos serviços fornecidos pelas várias camadas da pilha protocolar (Bellovin 1989).

Como já referido, é na camada de rede que se encontram os vários protocolos de encaminhamento interno e externo. Estes protocolos apresentam algumas vulnerabilidades que exploradas podem ter consequências graves no devido funcionamento da rede. Um exemplo disto é a propagação de falsas rotas e informação falsa do estado de ligação (Bruhadeshwar *et al.* 2009). Já na camada de transporte, onde se encontram os protocolos TCP e UDP, existem também alguns problemas de segurança. Estes dois protocolos baseiam-

²Imagem retirada em Junho de 2013 de <http://www.opte.org/maps/>

se no endereço IP para identificar os vários *host* da rede, não existindo assim nenhuma forma de autenticação, tanto ao nível da origem dos pacotes bem como da integridade e validade dos dados que são transferidos (Hao-yu *et al.* 2010).

Observando alguns protocolos comuns da camada de aplicação da pilha protocolar TCP/IP, tais como DNS (Ariyapperuma em Mitchell 2007), FTP (Xia *et al.* 2010), SNMP (Jiang 2002) e SMTP (Takesue 2011) (Riabov 2007), são relatados alguns problemas de segurança, como por exemplo, ao nível da autenticação, *sniffing*, *cache poisoning*, *stack overflow*, *spoofing* e negação de serviço.

2.3 Segurança em Redes de Computadores

Actualmente a expansão da *Internet* é um dos maiores fenómenos observados nos últimos anos. Vivemos numa sociedade cada vez mais dependente da "Rede", tanto a nível pessoal como empresarial. Sem grande "regra" a *Internet* cresce de dia para dia "tomando conta" do nosso quotidiano e dela somos, de certo modo, cada vez mais dependentes. Todo este rápido processo de popularização da *Internet*, bem como o facto de ser "aberta", partilhada e de dimensão mundial, faz com que seja cada vez mais vulnerável (Li 2012).

O conceito de Segurança em redes de computadores surge com a tentativa de eliminar, ou pelo menos atenuar as consequências dos problemas referidos, através da inclusão de medidas técnicas e da correta gestão da infraestrutura de rede. Este conceito assenta no estudo e compreensão de áreas como as ciências da computação, tecnologias de comunicação, redes de computadores, segurança da informação, teoria da informação e caracteriza-se pela privacidade, integridade, confiabilidade, disponibilidade e não repúdio de transferência de informação (Li 2012) (Yan-ping *et al.* 2010).

Existe um conjunto de problemas e ameaças que levam a ter em atenção o "assunto" da segurança das redes de computadores e da informação. Desde os problemas de *design* e implementação dos protocolos utilizados nas redes, passando pelos problemas de autenticação dos mecanismos de controlo e encaminhamento de rede, que são muitas vezes mínimos ou inexistentes. No entanto existem outras preocupações, nomeadamente todas as vulnerabilidades do vasto número de *software* utilizado, bem como toda a espécie de software malicioso existente. Estas são algumas razões fortes pela qual a segurança de redes de computadores e da informação é importante

(Xiaoling 2011) (Yan-ping *et al.* 2010).

Segundo Zúquete (2010), uma das grandes áreas de actividade que se devem considerar no âmbito da segurança computacional é a defesa contra actividade não autorizada e maliciosa. A defesa contra actividade não autorizada e maliciosa consiste na iniciativa de alguém contra o normal funcionamento dos sistemas computacionais. Este problema tem normalmente duas origens diferentes: indivíduos que pertencem à organização proprietária do sistema computacional ou indivíduos que não pertencem à organização.

As actividades não autorizadas dividem-se em cinco tipos diferentes:

- **Acesso a informação** - Todos os tipos de acesso a informação confidencial, quer se encontre guardada em sistemas computacionais ou se encontrem em trânsito na rede.
- **Alteração de informação** - Tudo que envolva a alteração e destruição de informação sem autorização, quer de forma implícita ou explícita.
- **Utilização abusiva de recursos** - O uso abusivo de recursos como a memória principal ou secundária, tempo de processamento e ocupação de rede, pode ter o efeito de não disponibilidade desses recursos a terceiros.
- **Impedimento de prestação de serviço** - Pode ser entendido como um caso extremo de utilização abusiva de recursos, pois normalmente o seu principal objectivo é impedir que terceiros tenham acesso aos recursos.
- **Vandalismo** - Consiste na interferência com o normal funcionamento dos sistemas computacionais, sem que isto traga qualquer benefício para quem o causa.

2.3.1 Detecção de intrusões

Entende-se por detecção de intrusões todos os processos de identificação de actividades não autorizadas e maliciosas, internas ou externas, numa dada rede de computadores. Uma intrusão pode ser definida por uma acção ou conjunto de acções que visam comprometer de alguma forma, um ou mais recursos de uma rede de computadores.

No âmbito da segurança da informação, a detecção de intrusões tem um

papel fundamental, onde os mecanismos básicos de segurança tais como: autenticação, controlo de acesso, *firewalls* e anti-vírus, por si só já não são suficientes para garantir a segurança dos sistemas computacionais, bem como toda a informação neles contida (Li *et al.* 2005).

Existe um conjunto de tecnologias que podem ser implementadas de forma a auxiliar e agilizar a detecção de intrusões e a actividade maliciosa. Das mais conhecidas e utilizadas destacam-se as *firewalls*, os IDS (Intrusion Detection System) e os anti-vírus. Embora todas estas tecnologias assumam o seu papel na detecção de actividade não autorizada, cada uma delas tem uma finalidade diferente em relação às demais.

Uma *firewall* consiste numa tecnologia que implementa um conjunto de medidas e políticas de segurança e controlo de acesso, entre duas redes com níveis de confiança diferentes, podendo ser vista como um "separador" ou um "limitador" entre duas redes. As *firewalls* impedem o acesso não autorizado, tanto a informação como a recursos do sistema, tendo um papel reactivo no que toca à segurança do sistema. Funcionalmente uma *firewall* analisa e filtra o tráfego da rede com base num conjunto de regras previamente estipuladas, seja para controlar o acesso dos *hots* da rede interna para a rede externa ou restringir o acesso da rede externa aos *hots* da rede interna (Yue *et al.* 2009).

Os anti-vírus são uma tecnologia que visa a protecção de sistemas computacionais contra vírus informáticos³, permitindo a sua detecção e eliminação. Esta tecnologia tem então a capacidade de ao detectar algum tipo de *malware* gerar um alerta e eliminar de alguma forma o problema tendo, como as *firewalls*, um papel activo em relação à segurança do sistema (Yue *et al.* 2009).

Os IDS são sistemas que têm como função detectar actividade classificada como fora do normal. Baseiam-se no principio de que o comportamento malicioso e não autorizado de um sistema computacional é claramente diferente do comportamento normal. De forma a detectar actividade maliciosa, os IDS analisam diversas fontes de dados do sistema computacional que está a ser monitorizado, detectando padrões anormais gerados pela actividade maliciosa (Zeng *et al.* 2010).

Os IDS são classificados em dois tipos: os IDS que operam num único *host* (HIDS) e os IDS que operam como um dispositivo autónomo numa rede

³*Software* malicioso que tal como um vírus biológico tem a capacidades de se propagar.

de computadores (NIDS). Os HIDS monitorizam os diversos componentes relativos a um *host* ao qual este se encontra hospedado, tais como: sistema operativo, sistema de ficheiros, *hardware*, tráfego de entrada e saída. Este tipo de IDS utiliza também os recursos disponíveis do *host* em que se encontra hospedado, para desempenhar a sua função. Em relação aos NIDS, estes monitorizam todo o tráfego de uma rede, podendo assim monitorizar um vasto número de *hosts*, de forma a detectar actividades ilícitas (Garuba *et al.* 2008).

Existem dois métodos de detecção de intrusões: SIDS (*Signature Based*) e AIDS (*Anomaly Based*). O primeiro método, contém uma base de dados de assinaturas de ataques conhecidos e analisa todo o tráfego da rede comparando-o com essas assinaturas. Caso encontre uma correspondência é gerado um alerta (Gupta *et al.* 2012). O segundo método consiste na detecção de desvios à normalidade da actividade, gerando um alerta caso tal se verifique.

Contudo os diferentes modos de detecção dos IDS têm alguns problemas e limitações. Os SIDS apenas detectam intrusões para as quais possuem assinatura e geram uma quantidade de falsos positivos elevada. Estes problemas podem ocorrer, por exemplo, devido à especificidade das assinaturas ou ao não conhecimento do ambiente que o IDS está a monitorizar (Gupta *et al.* 2012). Quanto aos AIDS, que apesar de terem maior capacidade de detectar ataques desconhecidos, apresentam problemas na classificação do tráfego da rede, dada a complexidade de o classificar como normal ou anormal, dando origem a muitos falsos alarmes. Para além do mais é necessário ainda inspeccionar manualmente um alerta, de forma a poder identificar se um ataque desconhecido existe ou não (Sato *et al.* 2012).

Em relação aos tipos de análise, os IDS podem ser do tipo *Singular Based* ou *Collaborative Based*. O tipo *Singular Based* consiste num tipo de análise mais simplista, onde existe apenas um sensor como fonte de dados. Já o tipo *Collaborative Based*, consiste na análise diversas fontes de dados (múltiplos sensores) provenientes de diversas camadas do sistema. Segundo Zhou *et al.* (2010), os *collaborative IDS*, ao contrário do tipo de análise *Singular Based*, têm a capacidade de reduzir o número de falsos positivos, devido ao facto de correlacionarem eventos provenientes de diversos sensores.

Na tabela 2.1 é possível visualizar de forma mais resumida a classificação dos IDS, em função das suas diferentes características.

| Característica operacional | Classificação |
|----------------------------|----------------------------|
| Modo de detecção | <i>Signature Based</i> |
| | <i>Anomaly Based</i> |
| Fonte de eventos | <i>Host Based</i> |
| | <i>Network Based</i> |
| Tipo de análise | <i>Singular Based</i> |
| | <i>Collaborative Based</i> |

Tabela 2.1: Classificação dos IDS: Modificado de (Zúquete 2010), Página 210

Os falsos positivos continuam a ser os maiores problemas, independentemente do tipo de IDS. Segundo Sourour *et al.* (2009) cerca de 99% dos alertas gerados pelos IDS não estão relacionadas com qualquer tipo de actividade maliciosa. O elevado número de falsos positivos leva à sobrecarga da entidade de segurança dada a quantidade de informação falsa a analisar (Shimamura en Kono 2006).

2.3.1.1 Registo de eventos de segurança

As tecnologias para detecção de intrusões e actividade maliciosa geram eventos quando detectadas anomalias ou quebras de segurança que podem comprometer o sistema a monitorizar. Dada a quantidade de eventos gerados por este tipo de tecnologias, é necessário um mecanismo de registo desses eventos, para posterior análise. Para tal são utilizados *logs*. *Logs* são compostos por entradas que contem informação relativa a um determinado evento que tenha ocorrido (Kent en Souppaya 2006).

As tecnologias acima referidas geram eventos quando detectam actividade anómala, contudo existem algumas particularidades que diferenciam os eventos gerados por estas. Os eventos gerados pelas *firewalls* e anti-vírus reportam uma quebra de segurança no sistema, já os eventos gerados pelos IDS reportam anomalias e actividade suspeita (Kent en Souppaya 2006). Este é o factor de diferenciação entre os eventos gerados por estas tecnologias, dado que a informação contida nos eventos gerados pelos IDS não é explícita

em relação à causa da sua geração. Por este motivo os eventos provenientes dos IDS requerem informação complementar de forma a enriquecê-los, agilizando todo o processo da sua análise.

Para caracterizar melhor um evento de segurança, uma das informações necessárias é a natureza do ataque que lhe estará associado. Nesse sentido a secção seguinte resume alguns dos aspectos fundamentais dos ataques e dos repetidos modelos de representação.

2.3.2 Modelação de ataques

Os ataques cibernéticos são a principal razão para o uso de políticas e mecanismos de segurança. Os ataques cibernéticos aumentam de dia para dia, são cada vez mais difíceis de detectar, tendo sempre um objectivo final e de natureza diversa. Os atacantes seguem uma sequência lógica de procedimentos para atingir o seu objectivo final, com recurso a ferramentas para o efeito, sendo possível visualizar na figura 2.3 as etapas necessárias na execução de um ataque genérico (Eom *et al.* 2008) (?).

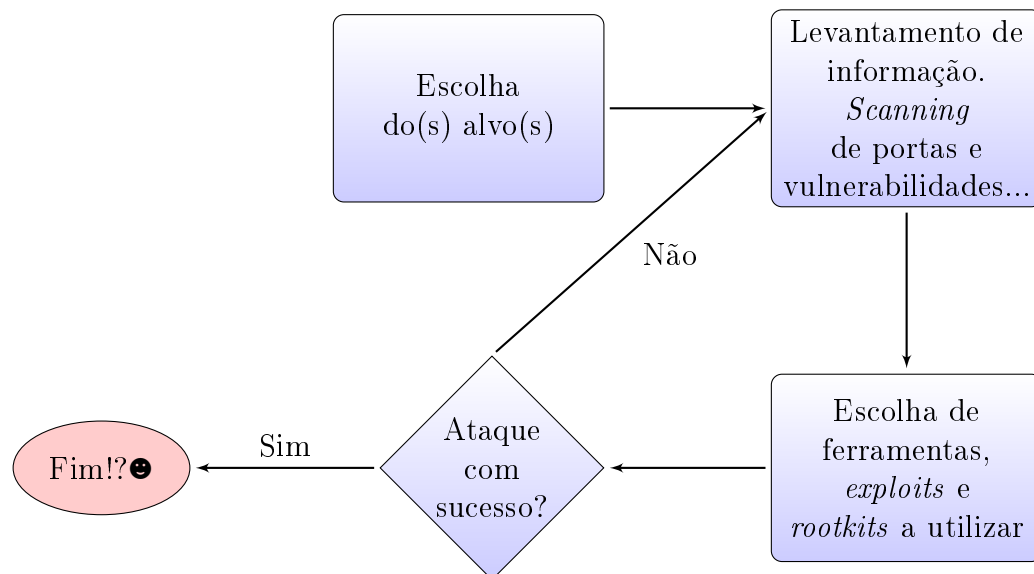


Figura 2.3: Etapas de um ataque

As técnicas utilizadas em ataques cibernéticos são cada vez mais robustas e dinâmicas, sendo cada vez mais difícil para os mecanismos de segurança informática detectarem estes ataques. Dado isto são necessários métodos

capazes de detectar os ataques em si e de ter a capacidade de avaliar os efeitos que estes causam, obtendo uma análise profunda e concisa sobre o impacto do ataque em relação ao sistema a proteger (Ning *et al.* 2008).

2.3.2.1 *Attack trees*

Attack trees consistem numa representação formal de como um sistema pode ser atacado, representando ataques numa estrutura em árvore composta por vários nós, que representam as várias formas de atingir a meta, sendo esta representada pelo nó da raiz da árvore (Camtepe en Yener 2007). Ao conseguir visualizar e compreender as diferentes maneiras que um sistema pode ser atacado, é possível conceber medidas de forma a tentar impedir esses ataques. Segundo Eom *et al.* (2008) cada nó tem três atributos: condições prévias, que são propriedades do sistema que facilitam a execução do ataque, sub-metas, que são os nós adjacentes ao nó da raiz da árvore e pós-condições, que são as mudanças em diferentes sistemas e ambientes. Segundo Camtepe en Yener (2007) é possível ainda capturar e identificar etapas atómicas de um ataque, sendo possível parametrizar custos e pesos a cada etapa do ataque.

Segundo Saini *et al.* (2008) O processo de construção uma *Attack tree* é a seguinte:

1. Definir o principal objectivo do ataque.
2. Decompor o objectivo do ataque em sub-objectivos.
3. Decompor gradualmente os sub-objectivos em em tarefas cada vez menores.
4. Atribuir valores aos atributos relativos aos nós adjacentes.
5. Atribuído os valores a todos os nós, é possível avaliar a segurança do sistema em relação ao objectivo do ataque.

Na figure 2.4 é possível visualizar um exemplo de um *attack tree*.

As *attack tress* apresentam contudo alguns problemas. Segundo Camtepe en Yener (2007) um ataque pode exigir que as suas acções atómicas aconteçam por uma dada ordem temporal rigorosa, ou então, que essas mesmas acções apenas são válidas num determinado período de tempo, o que é difícil de representar em *attack trees*. Segundo Saini *et al.* (2008) outro aspecto problemático, é o facto de não existir nenhum método sistemático para a atribuição dos valores aos atributos dos nós adjacentes da árvore.

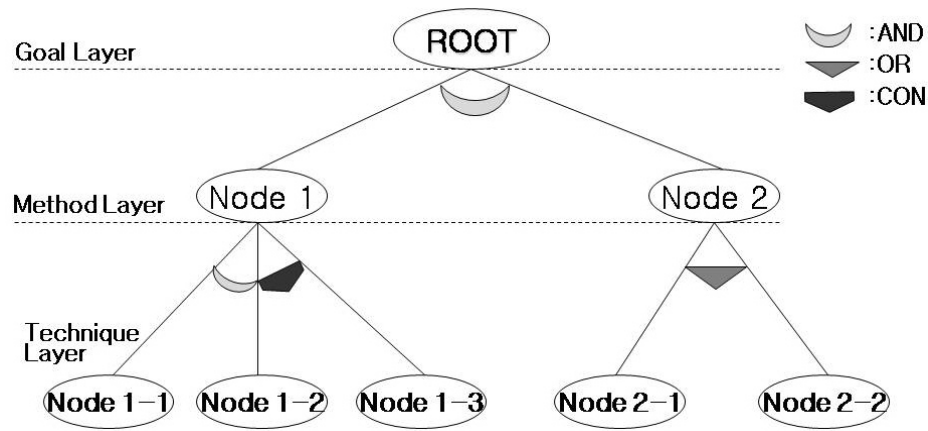


Figura 2.4: Exemplo de *Attack Tree*: Imagem retirada de (Eom *et al.* 2008)

2.3.2.2 *Attack graphs*

Attack graphs tal como *attack trees*, consistem numa representação formal de como um sistema pode ser atacado, bem como os processos e acções a que um atacante recorre para explorar vulnerabilidades de um sistema. São grafos direccionados onde podem ser descritos as possíveis acções tomadas por um intruso, bem como vulnerabilidades e *exploits*⁴ de um dado sistema. Para além disso, os *attack graphs* podem auxiliar na análise do número de caminhos de um ataque, bem como no cálculo de número de ataques bem sucedidos.

Segundo Alhomidi em Reed (2012) existem vários modelos de representação de *attack graphs*, com características e finalidades diferentes, sendo eles:

- ***Graph-Based Attack Graph***: este modelo pode ser considerado o modelo genérico ou modelo base, dos *attack graphs*. Ele demonstra como construir diagramas que representam ataques, usando os vários caminhos e probabilidades de sucesso de um ataque como base de conhecimento. Ele representa os estados possíveis de um ataque e os *exploits* utilizados, em *hosts* específicos, bem como serviços que os *hosts* fornecem. Os nós do grafo deste modelo representam o estado de um ataque e as ligações entre nós representam um *exploit*
- ***State Enumeration Attack Graph***: este modelo consiste num controlo simbólico, onde é representada o estado de uma rede, bem

⁴conjunto de código ou comandos que se aproveitam de vulnerabilidades presentes em sistemas computacionais.

como as transições de estado. As entradas deste modelo são as vulnerabilidades de cada *host* da rede, as ligações de rede e as possíveis capacidades e recursos do atacante. Dado o objectivo do ataque, o modelo produz o "grafo de ataque" e verifica se a sequência de vulnerabilidades leva à concretização do ataque. Com isto, este modelo revela que vulnerabilidades devem ser corrigidas. Contudo este modelo apresenta uma limitação pois apresenta caminhos de ataques redundantes, o que resulta em cálculos desnecessários. Os nós do grafo representam o estado de uma rede e as ligações representam a transição de estados.

- ***Coordinated Attack Graph:*** este modelo consiste na representação de ataques em conjunto, isto é, para um ataque alcançar o seu objectivo, este necessita de um grupo de intervenientes coordenados. Neste modelo é representado um sistema e as várias acções do ataque. É utilizado o modelo matemático de máquina de estados finita para representar as várias acções conjuntas que levam ao objectivo do ataque. O problema deste modelo é a dimensão que o grafo gerado pode apresentar, dado ao número de estados que este pode conter. Os nós deste grafo representam o estado de um sistema e as ligações representam uma acção.
- ***Dependency Attack Graph:*** este modelo baseia-se numa relação causal de condições relativas à especificação e configuração de um sistema, isto é, um pré-requisito de um determinado ataque nunca é inválido pelo sucesso de um outro ataque. Esta abordagem melhorou significativamente a escalabilidade em relação a outros modelos. No grafo gerado por este modelo, os nós representam condições de estado das configuração de um sistema e as ligações representam a relação causal entre essas condições.
- ***Full and Predictive Attack Graph:*** este modelo começa por representar o *host* do atacante e todas as suas acções possíveis. De seguida é representada a acessibilidade de outros *hosts* em relação ao *host* do atacante. Sempre que um *host* é alcançável, é feita uma tentativa de comprometer o *host* em função das suas vulnerabilidades. Contudo esta abordagem tem alguns problemas, pois gera caminhos redundantes. Os nós do grafo gerado por este modelo representam *hosts* de uma rede e as ligações entre os nós representam vulnerabilidades.
- ***Host-Compromised Attack Graph:*** este modelo consiste num grafo

de ataques construído com base em informação produzida por um teste de penetração, de forma a representar o estado de uma rede e os respectivos *exploits* utilizados. Este modelo indica que *hosts* podem estar comprometidos sem o conhecimento das acções do ataque utilizado. Os nós do grafo gerado por este modelo representam o estado de uma rede e as ligações representam a utilização de um *exploit*.

- **Topological Vulnerability Attack Graph:** este modelo consiste na visualização de uma espécie de topologia de vulnerabilidades, isto é, ele tem como objectivo a análise de vulnerabilidades de um sistema, como podem ser exploradas cada uma delas e quais as dependências entre si. O modelo cria um grafo de dependência de um dado ataque com os *exploits* a utilizar. Dado uma condição inicial ele mostra todas as condições seguintes para o sucesso do ataque. Neste modelo os nós do grafo gerado representam condições e as ligações representam um *exploit*
- **Host-Based Network Attack Graph:** este modelo é constituído por três partes: perfis de ataques, descrição de *hosts*, as ligações de rede e vulnerabilidades e por fim um conjunto de regras e acções de ataque. O algoritmo usado neste modelo para gerar o grafo de ataque começa pelas vulnerabilidades de um *host* e de seguida usa os recursos desse mesmo *host* para atacar outros *hosts*, até o ataque atingir o seu objectivo. Este modelo mostra a sequência de ataques a partir de um *host* origem até um *host* destino. Neste grafo os nós representam um ou mais *hosts* e as ligações representam um *exploit*.
- **Multiple-Prerequisites Attack Graph:** neste modelo estão presentes três tipos de nós: os estados, pré-requisitos e vulnerabilidades. Os estados representam *hosts* e o nível de acesso do atacante, os pré-requisitos representam as condições que um ataque necessita para conseguir explorar uma vulnerabilidade. No grafo gerado por este modelo, os nós podem representar um estado, um pré-requisito ou uma vulnerabilidade, e as ligações representam a transição entre os vários tipos de nós.
- **Logical Attack Graph:** este modelo baseia-se na relação de causalidade entre as configurações do sistema e os privilégios do atacante. São representadas as instruções lógicas de um ataque e as relações de causalidade da rede com os privilégios do atacante. O objectivo do ataque é representado no nó raiz do grafo gerado sendo o ataque expresso numa formula proposicional em relação aos

parâmetros de configuração da rede. Neste modelo os nós representam uma instrução lógica, enquanto as ligações representam a relação causal entre as configurações de rede e os potenciais privilégios do atacante.

- **Goal-Oriented Attack Graph:** neste modelo são representadas vulnerabilidades e as acções para a execução de um ataque, independentemente do sistema em causa. Este modelo tem como objectivo descrever todos os passos de um atacante, em função de um conjunto de pré-condições e os *exploits* que podem ser utilizados para um determinado ataque atingir o seu objectivo. No grafo gerado por este modelo os nós representam vulnerabilidades e as ligações representam acções.

Na figura 2.5 é possível visualizar um exemplo de um *attack graph: Host-based Network Attack Graph*

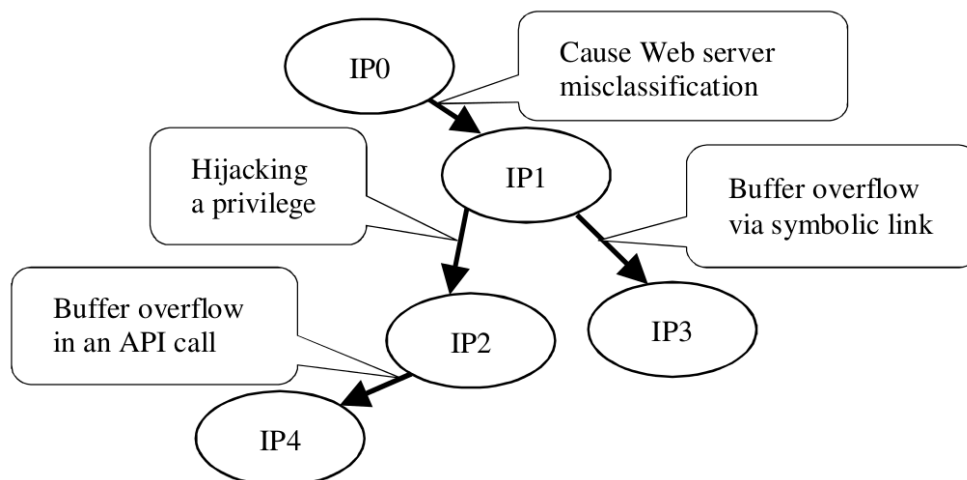


Figura 2.5: Exemplo de um *Attack Graph: Host-based Network Attack Graph*⁶

2.3.2.3 Common Attack Pattern Enumeration and Classification

Common attack Pattern Enumeration and Classification (CAPEC) (cap n.d.) é uma lista de padrões de ataques informáticos e uma descrição de métodos comuns de exploração de vulnerabilidade em sistemas de computação. Esta lista é de domínio público, desenvolvida por uma comunidade com o intuito

⁶Imagem retirada de (Alhomidi en Reed 2012)

de melhorar os aspectos de segurança no desenvolvimento de software e de ajudar à detecção e prevenção de ataques informáticos. Esta lista é uma espécie de "catálogo" e um esquema de classificação, que inclui uma taxonomia de classificação de ataques bem como a sua descrição, padrões e métodos comuns de exploração de vulnerabilidades, métodos e soluções para que de alguma forma o ataque seja evitado ou não surta efeito, fornecendo ainda uma perspectiva do atacante na óptica das suas possíveis motivações.

A lista de padrões de ataques do CAPEC segue uma estrutura em árvore para cada tipo de ataque, onde no topo da árvore se encontra a categoria dos ataques, seguido dos ataques subjacentes que pertencem a essa categoria. Cada ataque contém a seguinte informação (Barnum *et al.* 2007):

- ***Pattern Name and Classification*** - Identificador exclusivo do ataque.
- ***Attack Prerequisites*** - Condições/funcionalidades que devem existir para o sucesso do ataque.
- ***Typical Severity*** - indicador de severidade do ataque.
- ***Typical Likelihood of Exploit*** - Indicador de probabilidade de *exploit*.
- ***Description*** - Descrição do ataque.
- ***Related Vulnerabilities or Weaknesses*** - Referência às vulnerabilidades e fraquezas que o ataque explora.
- ***Method of Attack*** - Que tipo de ataque é usado, por exemplo: *Analysis, Brute Force, Spoofing...*
- ***Attack Motivation-Consequences*** - Resultado técnico desejado pelo atacante, para que este consiga alcançar o seu objectivo.
- ***Attacker Skill or Knowledge Required*** - Nível de conhecimento técnico que o atacante deverá ter para conseguir executar o ataque.
- ***Resources Required*** - Que recursos, do sistema computacional a atacar, são necessários para executar o ataque.
- ***Solutions and Mitigations*** - Que acções devem ser tomadas para mitigar o ataque.

- **Context Description** - Quais os contextos técnicos em que este ataque é relevante, como por exemplo o sistema operativo ou linguagem de programação.
- **References** - Outras fontes de informação disponíveis sobre o ataque.

Esta informação, depende da versão utilizada do CAPEC, sendo que o CAPEC está constantemente a ser actualizado pela comunidade que nela trabalha. Em alguns ataques nem sempre está disponível toda a informação aqui indicada.

2.3.3 Modelação de vulnerabilidades

Uma vulnerabilidade é uma falha na concepção, implementação ou operacionalidade de um sistema computacional, que pode ser explorada para violar um sistema computacional (IETF RFC 2828).

Diariamente são descobertas e reportadas novas vulnerabilidades, o que se traduz num aumento de "pontos fracos" nos sistema de computação, que leva ao aumento de ataques cibernéticos. Para além do mais, a relação temporal entre o anuncio de uma nova vulnerabilidade e o aparecimento de um ataque que explore essa mesma vulnerabilidade diminuiu, o que dificulta mitigação da ameaça (Cukier en Panjwani 2009).

CVE (cve n.d.) é um dicionário que contém informação sobre vulnerabilidades já conhecidas. Esta lista contém um nome para cada vulnerabilidade e a respectiva descrição. Tal como o CAPEC, o CVE é de domínio publico.

O processo de criação de um identificador CVE começa com a descoberta de uma potencial vulnerabilidade, sendo-lhe atribuído um identificador temporário CVE (Ex. *CVE-2000-0004*), ficando de seguida sujeito a um processo de verificação. Terminado o processo de verificação o estado da vulnerabilidade passa a definido. O CVE é hoje visto como um *standard* compatível com inúmeros serviços e produtos de segurança da informação.

2.3.4 *Security Information and Event Management*

Security Information and Event Management, ou simplesmente SIEM, consiste na combinação de *Security Information Management* (SIM) com *Security Event Management* (SEM). Esta abordagem foca-se na recolha e análise da informação de segurança relevante e proveniente de diversos

pontos, de forma a ser plausível a observação dessa informação de um único ponto de vista, mais amplo e específico.

Os sistemas SIEM fazem a recolha da informação gerada pelos vários sensores do sistema computacional, como por exemplo IDSs e *firewalls*, correlacionando toda essa informação, de forma a conseguir detectar intrusões. Os SIEM tiram partido de todo histórico de informação, de forma a otimizar a precisão de detecção de intrusões e anomalias, com a criação de modelos dos mesmos. Na figura 2.6 é possível visualizar a arquitectura conceptual de um SIEM (Gabriel *et al.* 2009) (Souppaya *et al.* 2006).

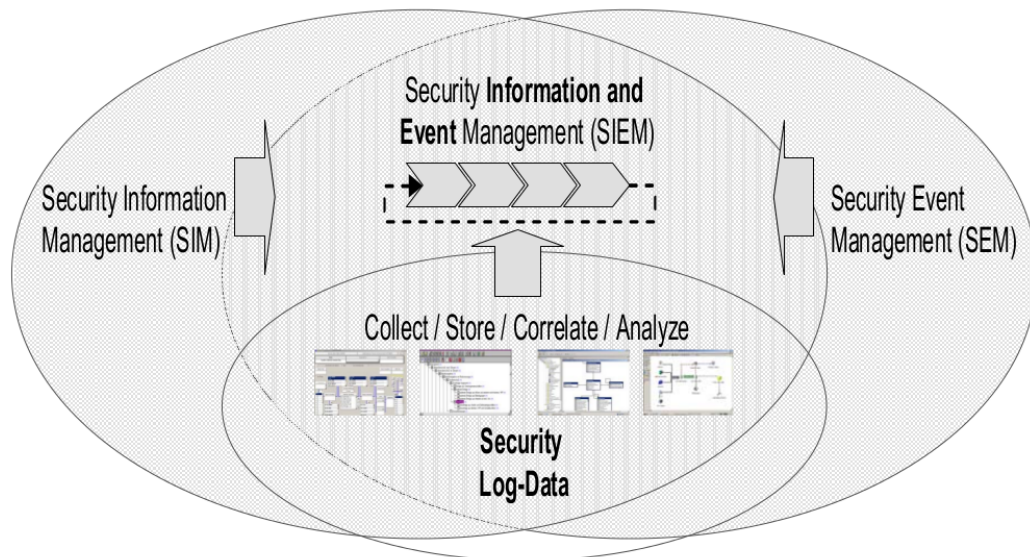


Figura 2.6: Arquitectura conceptual dos SIEM: Imagem retirada de (Gabriel *et al.* 2009)

Os SIEM podem fazer a recolha de eventos utilizando um de dois métodos: *Agentless* ou *Agent-Based*. No caso de *Agentless* o servidor SIEM recebe os dados dos vários *host* que estão a ser monitorizados e de seguida são efectuadas as funções filtragem, agregação, normalização e análise dos eventos. Em relação ao método *Agent-Based*, é instalado um *software* em cada *host* a monitorizar, de forma a este efectuar as tarefas de filtragem, agregação e normalização de eventos, para de seguida enviar para ao servidor SIEM, para que este proceda à sua análise e armazenamento (Souppaya *et al.* 2006).

Segundo Miller *et al.* (2010) um SIEM é normalmente composto por seis partes:

- **Source Device** - Fonte de onde provém toda a informação a ser analisada pelo Servidor SIEM. A informação pode ter origem de qualquer equipamento real, como um *router*, bem como ter origem dos *logs* de uma aplicação.
- **Log Collection** - Consiste na recolha da informação gerada pelos vários *Source Device*.
- **Parsing/Normalization** - Neste processo toda a informação recolhida é normalizada para um formato padrão.
- **Rule Engine/Correlation Engine** - O *Rule engine* é o mecanismo responsável por gerar alertas em função da informação contida nos eventos. O *Correlation Engine* é um sub processo do *Rule Engine*, que tem como função correlacionar os vários eventos de diferentes origens.
- **Log Storage** - Processo responsável por armazenar toda a informação. Os SIEM podem guardar a informação numa base de dados, num simples ficheiro de texto ou em ficheiro binário.
- **Event Monitoring** - Interface com o utilizador. Onde é apresentada toda a informação processada pelo SIEM. Onde é efectuada toda a configuração do SIEM.

Na figura 2.7 é possível visualizar a arquitectura de um SIEM:

2.4 Ferramentas/Frameworks

As ferramentas de segurança têm hoje um papel importante na tentativa de salvaguardar as propriedades de segurança dos sistemas computacionais. Existem ferramenta de diversos tipos com propósitos e funcionalidades diferentes, seja na detecção de intrusões ou na verificação de falhas e vulnerabilidades, tendo como objectivo auxiliar todos os processos que envolvem a segurança da informação.

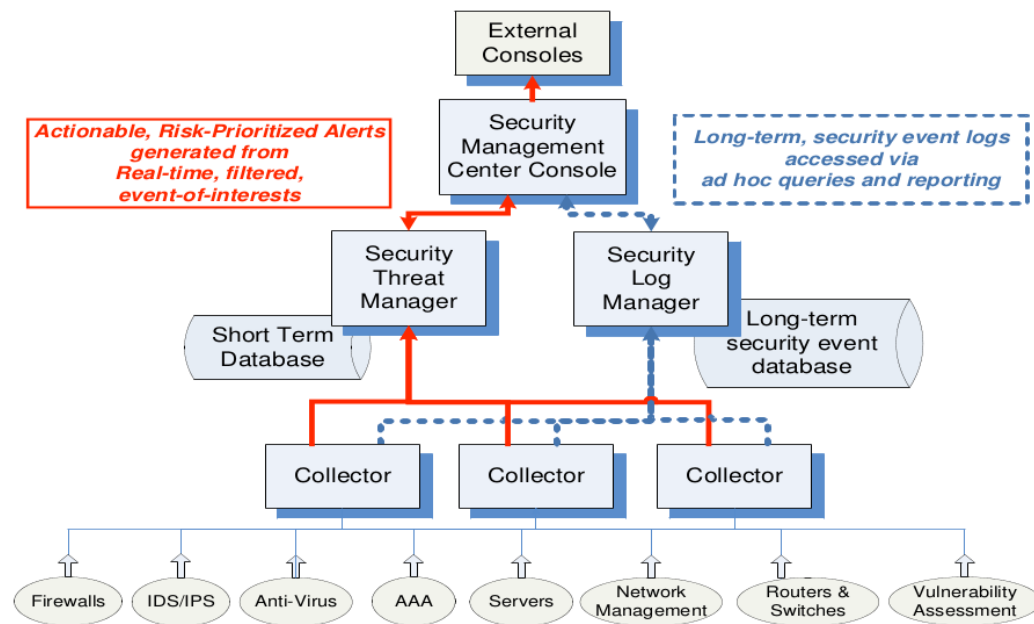


Figura 2.7: Arquitectura genérica dos SIEM: Imagem retirada de (Swift 2006)

2.4.1 Snort

O *Snort* é um detector de intrusões utilizado na inspecção de redes de computadores, que possui um motor de monitorização de tráfego de rede. é um NIDS baseado em conhecimento multiplataforma, que contém um vasto número de assinaturas de ataques conhecidos. É normalmente utilizado na monitorização de redes de pequena e média escala, permite a detecção de ataques em tempo real e a adição de módulos de resposta a ataques (Roesch 1999a).

Ao nível da arquitectura, o *Snort* é composto por três camadas funcionais: o decodificador de pacotes, o mecanismo de detecção e o mecanismo de registo e alerta. Estas camadas assentam sobre a biblioteca *libcap*, que permite a captura de datagramas IP em diversos sistemas operativos sendo usada também por várias aplicações de análise de tráfego. A decodificação de pacotes fornece informação sobre os datagramas IP. Os mecanismos de detecção são compostos por um conjunto de pré-processadores, que fazem uma análise prévia e uma normalização da informação capturada, que incluem módulos de reconstrução de datagramas, detecção de actividades em portos da camada de transporte, detecção de tráfego anormal e normalizadores de protocolos de camada de aplicação. Por fim, o mecanismo

de registo e alerta, baseado em regras pré-definidas, em que cada regra específica um conjunto de padrões característicos de um datagrama de um ataque previamente conhecido. Todas as regras activas são percorridas, até encontrar uma que se enquadre com os dados a analisar, de forma a poder tomar uma decisão. Essa decisão pode ser aceitar os dados como normais, registar apenas a sua ocorrência ou gerar um alerta (Roesch 1999a) (Roesch 1999b).

Na figura 2.8 é possível visualizar a arquitectura do *snort*.

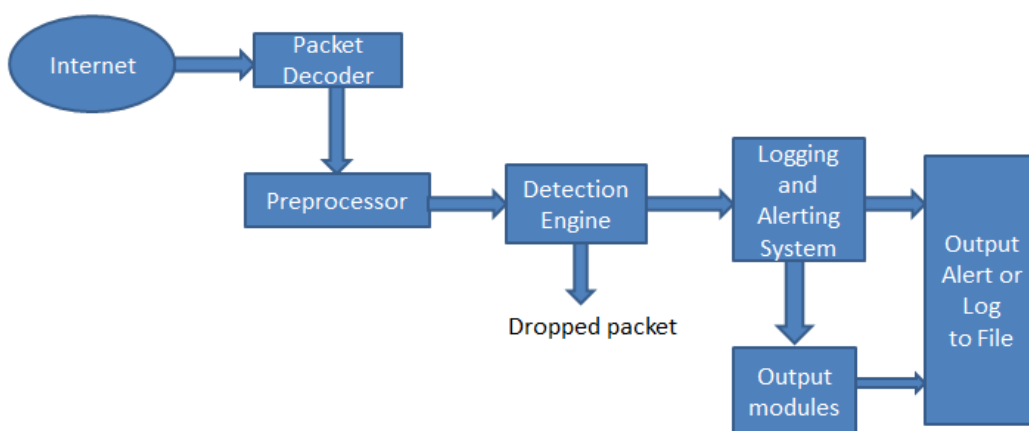


Figura 2.8: Arquitectura do *Snort*⁷

2.4.2 *Ossec*

O *Ossec*⁸ é um HIDS com a capacidade de análise de *logs*, verificação da modificação não autorizada de ficheiros, detecção de actividade maliciosa e monitorização de políticas de segurança. É uma ferramenta multiplataforma com capacidade de gerar alertas em tempo real e com a capacidade de resposta a ataques.

O *Ossec* segue uma arquitectura centralizada, permite configurar as políticas de segurança bem como configurar a prioridade de cada tipo de alerta, que contem um bloco de integração com o protocolo SMTP, que permite gerar notificações de alertas via *e-mail*. O *Ossec* é composto por dois blocos fundamentais: o *Manager* e os *Agents*. O *Manager* é o bloco central da sua arquitectura, é onde são armazenados os *logs*, as base de dados da

⁷Imagem retirada em Junho de 2013 de <http://seclists.org/>

⁸<http://http://www.ossec.net/>

visitado em Junho de 2013

Modelo para definição de criticidade em eventos de segurança de redes de computadores

verificação da integridade dos ficheiros e todos os eventos *Ossec*. São também armazenadas no *Manager* as configurações, as regras e políticas de segurança, bem como os decodificadores, o que faz com que a configuração do *Ossec* seja relativamente fácil.

Para além do *Manager* o *Ossec* contém outro bloco, os *Agents*, que consiste em *software* instalado nos *hosts* a serem monitorizados. O *agent* tem como função recolher toda a informação do *host* a analisar, enviando de seguida para o *Manager* para ser devidamente analisada e correlacionada. A ferramenta *Ossec* permite ainda a instalação de *Agents* em máquinas virtuais e permite ainda a recolha, e posteriormente análise, de informação proveniente de *routers*, *switches* e *firewalls*.

Na figura 2.9 é possível visualizar a arquitectura do *Ossec*.



Figura 2.9: Arquitectura do *Ossec*⁹

⁹Imagem retirada em Junho de 2013 de <http://www.ossec.net/>

2.4.3 *Open Source Security Information Management*

OSSIM (*Open Source Security Information Management*)¹⁰ é um SIEM desenvolvido pela *AlienVault*¹¹ com a capacidade de correlação e normalização de eventos de segurança. O *OSSIM* agrega um conjunto de ferramentas de segurança de forma a obter informação de segurança dos mais diversos tipos e fontes.

Como é possível verificar na figura 2.10 o *OSSIM* é composto por quatro blocos fundamentais: os sensores, o *Management Server*, a base de dados e o *Frontend* (Madrid *et al.* 2009).

- **Sensores** - Fazem a recolha da informação de segurança nos diversos pontos da rede, de forma passiva para não afectarem o tráfego da rede. Os sensores tem como função a recolha de informação tais como: intrusões, vulnerabilidades, anomalias, monitorização de redes. Para além de isso, os sensores têm ainda a capacidade de colher informação de *routers*, *firewalls* e outros IDSs. O *agent*, que está incorporado nos sensores, é responsável por enviar a informação dos sensores para o *Management Server*.
- **Management Server** - É onde a informação é tratada, analisada e correlacionada. É no *Management Server* que estão definidas as políticas de segurança. O *Management Server* é composto por um *daemon*¹² de controlo das partes integrantes do *OSSIM* e pelo *OSSIM Server* que centraliza toda a informação recebida dos sensores.
- **Base de dados** - Contêm os eventos e toda a informação útil para a gestão do sistema.
- **Frontend** - É responsável por toda a visualização e interacção da aplicação com o utilizador, sendo no *Frontend* que são efectuadas todas as configurações do *OSSIM*.

¹⁰<http://communities.alienvault.com/>

visitado em Junho de 2013

¹¹<http://www.alienvault.com/>

visitado em Junho de 2013

¹²Programa que corre em *Background*, sem que o utilizador tem qualquer interacção e controlo directo sobre ele

¹³Imagem retirada em Junho de 2013 de

<https://www.alienvault.com/wiki/doku.php?id=documentation:architecture>

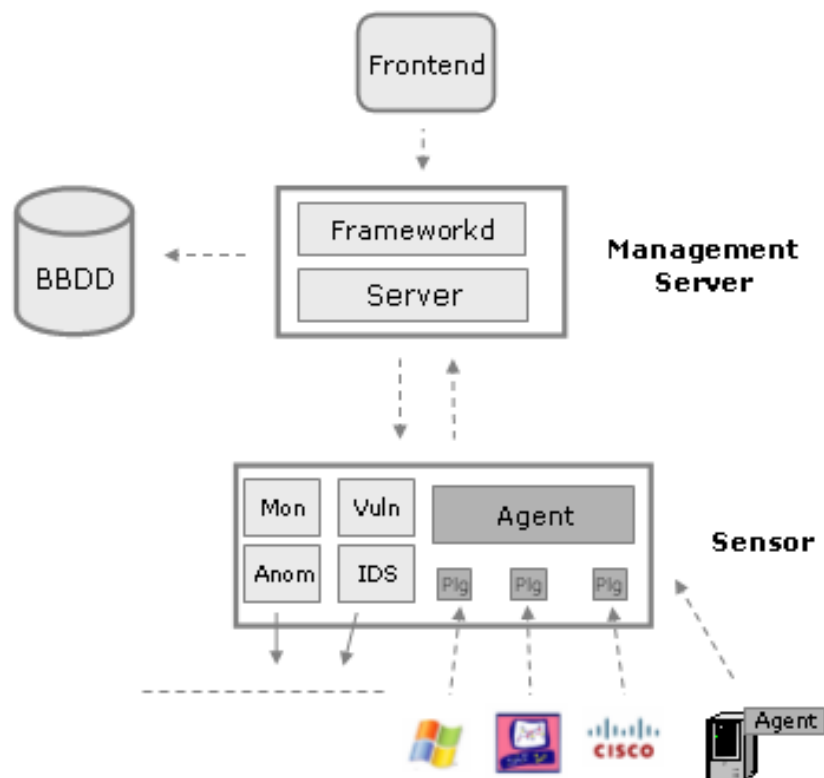


Figura 2.10: Arquitectura do *OSSIM*¹³

2.4.4 *Bro*

Bro, é um NIDS que analisa o tráfego de rede à procura de actividade maliciosa. Para além de ser um NIDS, *Bro* suporta uma variedade de funcionalidades no campo da análise de tráfego. Uma das principais características desde NIDS, é um facto de conter um extenso conjunto de ficheiros *logs*, onde são registadas todas as actividades da rede monitorizada em todos os níveis. O *Bro* faz o registo das várias ligações de rede, bem como de toda a actividade da camada de aplicação, tais como: sessões HTTP, solicitações de DNS, certificados SSL, etc.

Ao nível da arquitectura o *Bro* é fundamentalmente composto por dois componentes principais: o *event engine* e o *policy script interpreter*. O *event engine* tem como finalidade analisar todo o tráfego da rede, criando eventos de alto nível através do fluxo de pacotes onde os eventos não representam obrigatoriamente intrusões, mas sim a actividade da rede. O *event engine* começa por analisar os cabeçalhos dos pacotes de forma a verificar se estes

estão bem formados. Caso a verificação falhe o *Bro* gera um alerta indicando o problema e descarta o pacote, caso contrário, são gerados eventos de alto nível, e posteriormente enviados para o *policy script interpreter*. O *policy script interpreter* tem como função analisar esses eventos com base num conjunto de políticas de segurança, de forma a identificar tráfego malicioso. O *policy script interpreter* contém um conjunto de *Scripts* que analisam os eventos gerados pelo *event engine*, de forma a gerar alertas de segurança e/ou tomar uma atitude activa quando encontrada uma intrusão (Paxson 1998).

Na figura 2.11 é possível visualizar a arquitectura do *Bro*.

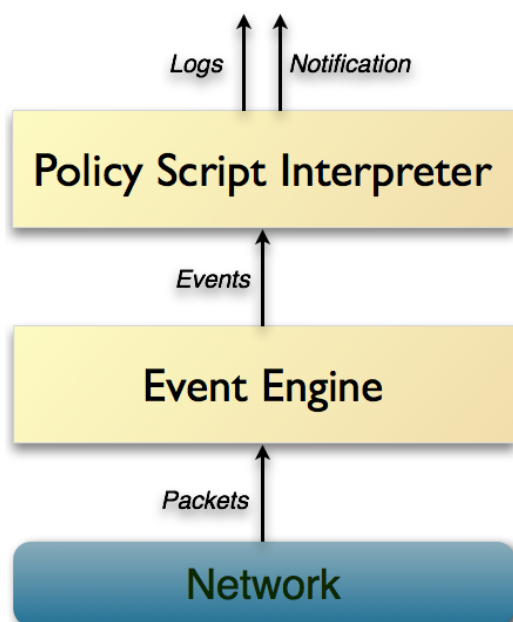


Figura 2.11: Arquitectura do *Bro*¹⁴

¹⁴Imagem retirada em Junho de 2013 de <http://www.bro.org/documentation/overview.html>

Capítulo 3

Especificação e implementação do sistema

No presente capítulo é apresentada a descrição da arquitectura da solução proposta, bem como todas as suas funcionalidades de forma a satisfazer os objectivos propostos. São também descritas todas as decisões tomadas relativamente à implementação do sistema proposto e a descrição das características do mesmo.

Inicialmente são especificados os requisitos do sistema seguido do desenho e arquitectura do mesmo, devidamente explicada e fundamentada. Por fim é feita uma descrição detalhada da implementação do sistema, de forma a salientar todas as opções tomadas que derivam dos requisitos inicialmente descritos.

3.1 Requisitos do sistema

A especificação dos requisitos do sistema é uma tarefa importantíssima, de forma a poder identificar as necessidades funcionais e não funcionais do mesmo. Através do estudo e análise da informação resultante dos eventos de segurança gerados pelos IDS, tentou-se identificar quais os principais aspectos que poderiam, eventualmente, melhorar a análise dos eventos de segurança, uma vez que os IDS geram muitos falsos positivos. Para além dos mais, os eventos de segurança gerados pelos IDS são de difícil análise, em relação a outras tecnologias de segurança. Os vários cenários em que o sistema poderá actuar têm as suas diferenças, o que também foi levado em consideração na especificação dos requisitos, de forma a que o sistema seja moldável para

diferentes contextos.

Com a especificação dos requisitos funcionais identificou-se as características técnicas dos sistema, de forma a obter os resultados esperados, em função do objectivo proposto neste trabalho. Em relação à especificação dos requisitos não funcionais, que embora não estejam directamente relacionados com as funcionalidades e características técnicas do sistema, estes caracterizam a sua usabilidade. Os requisitos não funcionais¹, de uma forma geral, caracterizam o sistema quanto à sua disponibilidade, manutenção, portabilidade, eficiência e fiabilidade, por exemplo.

3.1.1 Cenários e contextos

Observando um ataque cibernético, independentemente do que o caracteriza, este tem diferentes consequências dependendo do cenário que foi exposto ao ataque. Os danos causados por um ataque têm maior ou menor relevo em função do conjunto de factores que caracterizam o cenário. Cada cenário tem as suas próprias características, daí quando se analisa um evento de segurança é necessário ter em conta os aspectos que levaram a geração do mesmo, dependendo do tipo e da importância da informação nele contida.

Diferentes tipos ataques fazem os IDS gerar diferentes eventos, daí a necessidade de os classificar e priorizar, isto dependendo sempre do cenário e contexto. A título de exemplo, observando todas as consequências que um ataque de *Denial of Service* (DoS) sobre um operador de telecomunicações, estas são bastante diferentes se o mesmo ocorresse na rede de um sistema de saúde. Observando as consequências de um ataque *Sql Injection*, em função dos cenários acima descritos, este à partida teria muito mais impacto na rede do sistema de saúde do que na rede de um operador de telecomunicações.

Todas estas premissas devem ser tomadas em consideração quando desenhada uma estratégia de segurança, de forma a aumentar a detecção precoce da possível actividade maliciosa que mais estragos poderá fazer em relação ao contexto analisado.

3.1.2 Requisitos funcionais

Para o problema identificado foram definidos os seguintes requisitos funcionais:

¹IEEE Std 830-1998

- Recolher e interpretar informação proveniente do IDS *Snort*.
- Relacionar eventos de segurança com informação de contexto.
- Classificar eventos de segurança quanto ao tipo de ataque que estes possam representar.
- Identificar se um evento de segurança é oriundo de uma origem reputado como malicioso.
- Identificar quais os ataques que um evento de segurança possa estar associado.
- Descrever toda a informação disponível de um ataque.
- Relacionar informação recolhida do sistema computacional a monitorizar, nomeadamente as suas vulnerabilidades, com informação recolhida do IDS.
- Descrever toda a informação disponível de uma vulnerabilidade.
- Classificar eventos de segurança quanto a sua perigosidade e criticidade.
- Priorizar eventos de segurança, em função de valores introduzidos pelo utilizador.

3.1.3 Requisitos não funcionais

Para o problema identificado foram definidos os seguintes requisitos não funcionais:

- Multi-plataforma.
- Escalável.
- Composto por módulos funcionais independentes entre si.
- Tolerante à possível falta de informação específica.
- Interface gráfica simples e intuitiva.
- Possibilidade de implementação de novas funcionalidades.
- Tempo de resposta do sistema o mais curto possível.

3.2 Modelo de classificação e valorização de eventos

Conhecido o problema bem como todas as razões e características que o classificam como tal, neste trabalho foi criado um sistema capaz de valorizar e classificar eventos de segurança de forma a priorizar a análise dos mesmos.

O eventos analisados são oriundos do IDS *Snort*, por isso optou-se por criar um sistema que opere como uma camada superior deste. O sistema faz a recolha periódica dos eventos gerados pelo *Snort* e de seguida analisa cada um dos eventos individualmente, de forma a valoriza-lo com informação de contexto, obtendo assim a respectiva criticidade do mesmo.

No sistema foram implementadas as seguintes funcionalidades:

1. Recolher eventos gerados pelo *Snort*.
2. Verificar as vulnerabilidades associadas a cada evento.
3. Relacionar as vulnerabilidades de um evento com as vulnerabilidades do sistema a monitorizar (*Match* de vulnerabilidades).
4. Criar lista de possíveis ataques que cada evento pode representar, tenham eles já ocorrido, ou apenas sejam uma possibilidade.
5. Calcular criticidade de cada evento.
 - Calcular perigosidade de cada evento.
 - Identificar o tipo de ataque que cada evento possa ter associado.
 - Reputação do endereço IP de origem de cada evento.

Dado que o *Snort* é um NIDS, os eventos gerados correspondem à rede a que está configurado para monitorizar.

3.2.1 Vulnerabilidades

A cada evento analisado existe um conjunto de vulnerabilidades associadas. Quando o IDS verifica actividade suspeita, gera um evento relativo à mesma, estando normalmente essa actividade associada a uma ou mais vulnerabilidades do sistema computacional. Cada regra do *Snort* contém os indicadores CVE relativos a cada vulnerabilidade, o que torna mais fácil

essa verificação.

Contudo essas vulnerabilidades podem ou não estar presentes no sistema computacional a monitorizar, o que à partida se traduz num evento de pouca importância, ou até mesmo inofensivo. Um ataque dirigido a um determinado sistema, que explora uma determinada vulnerabilidade que não se encontra presente nesse sistema, tem normalmente uma taxa de sucesso reduzida.

Para isso é necessário que o sistema conheça as vulnerabilidades do sistema computacional a monitorizar, de forma a ser possível relacionar as vulnerabilidades associadas a um evento com as vulnerabilidades que se encontram presentes no sistema.

3.2.1.1 Associação de vulnerabilidades

De forma a verificar as vulnerabilidades do sistema computacional a monitorizar, é necessário recorrer a ferramentas para o efeito. Para o efeito deste projecto foi utilizado o *software Nessus*². Este *software* efectua um *scan* a cada *host* da rede a monitorizar, fazendo correr um conjunto de *scripts* de modo a encontrar problemas de segurança. Estes problemas de segurança podem ser vulnerabilidades do sistema, vulnerabilidades de aplicações instaladas e erros de configuração. No final deste processo o *Nessus* gera um relatório contendo essa informação.

Com a informação das vulnerabilidades e problemas de segurança do sistema a monitorizar, é possível verificar quais vulnerabilidades correspondentes a cada evento que estão presentes em cada *host* da rede.

Esta associação de vulnerabilidades dos eventos e do sistema só é possível devido ao facto de os eventos gerados pelo *Snort* e a informação resultante do *Nessus* utilizarem a mesma nomenclatura. Tanto o *Snort* como o *Nessus* utilizam a lista CVE para identificarem as vulnerabilidades.

Através das vulnerabilidades catalogadas na lista CVE é possível aceder à descrição detalhada de cada vulnerabilidade, bem como saber quando esta foi reportada.

²*Software* que possibilita verificar as falhas e as vulnerabilidades de uma rede de computadores, efectuando um *scan* para o efeito.

3.2.2 Ataques

De forma a identificar os ataques que possam estar associados aos eventos de segurança gerados, foi adoptada uma estratégia que consiste na correspondência da assinatura do evento com a descrição do ataque. Esta correspondência é efectuada através de palavras chave contidas na assinatura do evento gerado pelo *Snort*. Com essas palavras foi possível criar um dicionário, de forma a ser possível associar as mesmas às descrições contidas nas lista CAPEC. Sempre que um ataque tem uma correspondência maior ou igual que 60% é adicionado à lista de ataques do evento em análise. Apesar de tudo esta solução apresenta as suas limitações, sendo estas descritas posteriormente. Dado isto, para cada evento analisado é criada uma lista de ataques que podem corresponder ao tipo de anomalia detectada.

3.2.3 Criticidade

No projecto desenvolvido a criticidade consiste na valorização e classificação de um evento, de forma a priorizá-lo em relação aos demais. Para a obtenção da criticidade de um evento foram definidos os seguintes parâmetros: a sua perigosidade, se o endereço consta na lista de reputação e pelo tipo de ataque mais relevante que este possa representar. Dado que um evento pode corresponder a vários tipos de ataques, o tipo de ataque mais relevante é aquele que for definido inicialmente como tal, através de parâmetros, onde é efectuada uma priorização dos diferentes tipos de ataques.

Na prática a criticidade de um evento consiste num número de 0 a 100, sendo este calculado em função destas três proposições.

3.2.3.1 Perigosidade

A perigosidade de um evento é um valor numérico que pode variar entre 0 e $+\infty$, que consiste num indicador de quão prejudicial pode ser a causa que levou a geração do evento, independentemente do sistema computacional de onde este foi gerado. Contudo, um evento pode obter um valor elevado na perigosidade, mas pode não representar uma ameaça crítica, ou ser considerado pouco crítico no sistema computacional em causa.

O valor da perigosidade é calculado através de informação relativa às vulnerabilidades e ataques associados ao evento, bem como de informação contida no relatório *Nessus*.

Relativamente às vulnerabilidades são utilizados os seguintes indicadores, provenientes da lista de vulnerabilidades CVE:

- **Status** - Indica se a vulnerabilidade em causa já faz parte da lista CVE (*Entry*) ou se ainda se encontra sobre análise (*Draft*).
- **Age** - Ano em que foi reportada a vulnerabilidade.

Relativamente aos ataques são utilizados os seguintes indicadores, provenientes da lista de ataques CAPEC:

- **Severity** - Severidade do ataque
- **Typical Likelihood of Exploit** - Dificuldade de realização e/ou concretização do ataque.
- **Attacker Skills or Knowledge Required** - Nível (genericamente) de perícia exigida ao atacante.
- **CIA Impact** - Impacto do ataque relativamente as propriedades de confidencialidade, integridade e disponibilidade.

Relativamente ao relatório *Nessus*, os indicadores utilizados para o cálculo da perigosidade são relativos à porta de destino indicada pelo evento, sendo estes:

- **Severity** - Severidade relativa ao problema encontrado e/ou relativa a vulnerabilidades no serviço activo na porta em questão.
- **Risk factor** - Factor de risco relativa ao problema encontrado e/ou relativo a vulnerabilidades no serviço activo na porta em questão.

Todos os indicadores, à excepção do indicador *Age* relativo às vulnerabilidades, são qualitativos e apresentam-se em níveis de *Very Low*, *Low*, *Medium*, *High* e *Very High*. Dado que a perigosidade é um indicador numérico, é efectuado um *parsing* para um valor numérico correspondente: *Very Low* - 0, *Low* - 1, *Medium* - 2, *High* - 4 e *Very High* - 5. Em relação ao indicador *Age*, dado que este representa a idade da vulnerabilidade, a operação de transformação é um pouco diferente. Primeiro verificou-se qual o ano da vulnerabilidade mais antiga presente na lista de vulnerabilidades (CVE). Sempre que usada uma vulnerabilidade com o

mesmo ano dessa vulnerabilidade, o indicador *Age* toma o valor de 0, e para vulnerabilidades seguintes o indicador toma o valor da diferença de idades entre a vulnerabilidade mais antiga e a vulnerabilidade em questão. Esta estratégia foi adoptada devido à proposição que quanto mais recente for a vulnerabilidade, menos informação existe sobre ela, sendo potencialmente mais perigosa em relação a vulnerabilidades mais antigas e bem conhecidas.

Em função de todos estes indicadores, e através de uma da experiência laboratorial realizada, a perigosidade de um evento é calculada através da equação (3.1) :

$$PER = PER_{vul} + PER_{att} + PER_{nessus} \quad (3.1)$$

$$PER_{vul} = Status + Age \quad (3.2)$$

$$PER_{att} = Severity + TypicalLikelihoodOfExploit + AttackerSkillsOrKnowledgeRequired + CIAimpact \quad (3.3)$$

$$PER_{nessus} = Severity + RiskFactor \quad (3.4)$$

O valor calculado através da equação de perigosidade tem um peso de 40% do calor da criticidade.

3.2.3.2 Tipos de ataque

Um evento de segurança é gerado quando detecta actividade suspeita ou algum tipo de anomalia, ou seja, actividade fora do normal que pode ou não representar actividade maliciosa. Hoje em dia os vários tipos de actividade maliciosa podem ser caracterizados e identificados num tipo específico de ataque cibernético. Os ataques cibernéticos possuem características muito específicas, que permitem caracterizar os mesmos quanto ao seu tipo em função dos seus efeitos e consequências. Contudo, quando um evento detecta actividade maliciosa, esta pode não ser correspondente apenas a um único tipo ataque, mas sim a vários tipos.

Existem alguns aspectos comuns em diferentes tipos de ataques, ou seja, quando um evento é gerado, este por vezes não é totalmente específico quanto ao tipo de *malware* ou anomalia. Dado que para um ataque cibernético atinja o seu objectivo, é necessário um conjunto de operações, o que faz com que os IDS gerem vários eventos relativos a esse ataque.

De forma a priorizar os vários eventos em função do tipo de ataque que levou à sua geração, é necessário atribuir pesos aos vários tipos de ataque. O peso de cada tipo de ataque é atribuído quando iniciado o sistema, sendo estes introduzidos como parâmetro. Para cada tipo de ataque é introduzido um valor de 0 a 100%. Na lista de ataques de cada evento é verificado qual o ataque que maior peso tem em função do seu tipo, sendo o peso desse ataque utilizado para o cálculo da criticidade. O valor do ataque com maior peso é multiplicado por 40%, sendo este o peso utilizado no cálculo da criticidade.

Os diferentes tipos de ataque são: *Data Leakage Attacks*, *Resource Depletion*, *Path Traversal*, *Injection (Injecting Control Plane content through the Data Plane)*, *Spoofing*, *Time and State Attacks*, *Abuse of Functionality*, *Functionality Misuse*, *Abuse of Communication Channels*, *Probabilistic Techniques*, *Fingerprinting*, *Exploitation of Authentication*, *Exploitation of Privilege/Trust*, *Privilege Escalation*, *Remote Code Inclusion*, *Data Structure Attacks*, *Resource Manipulation*, *Web Services Protocol Manipulation* *Physical Security Attacks*.

3.2.3.3 Reputação IP

A reputação IP consiste na averiguação se o endereço do evento gerado consta numa lista de endereços classificados como maliciosos. A lista de endereços maliciosos utilizada neste projecto é proveniente de um grupo de investigação na área da segurança da informação, denominado de *AlienVault*. Esta lista encontra-se disponível para livre utilização, fazendo parte de um projecto denominado de *Open Source IP Reputation Portal* ³ desse grupo de investigação.

Sempre que o endereço de um evento analisado se encontra na lista de reputação, a criticidade desse evento aumenta 20%.

3.3 Arquitectura e implementação do sistema

Na presente secção são apresentadas detalhadamente todas as decisões de arquitectura e implementação do sistema desenvolvido.

³<http://labs.alienvault.com/labs/index.php/projects/open-source-ip-reputation-portal/> visitado em Agosto de 2013

3.3.1 Arquitectura do sistema

O sistema desenvolvido é baseado numa arquitectura em módulos independentes entre si, como pode ser visualizado na figura 3.1, em cada módulo estão implementadas um conjunto de funções para resolver problemas de diferente natureza. A independência dos módulos permitiu que a implementação do sistema fosse simplificada e tornou o sistema mais flexível a possíveis alterações no futuro. Para além disso, a independência entre módulos permite a possibilidade de acrescentar novos módulos, de forma a aumentar as funcionalidades do sistema.

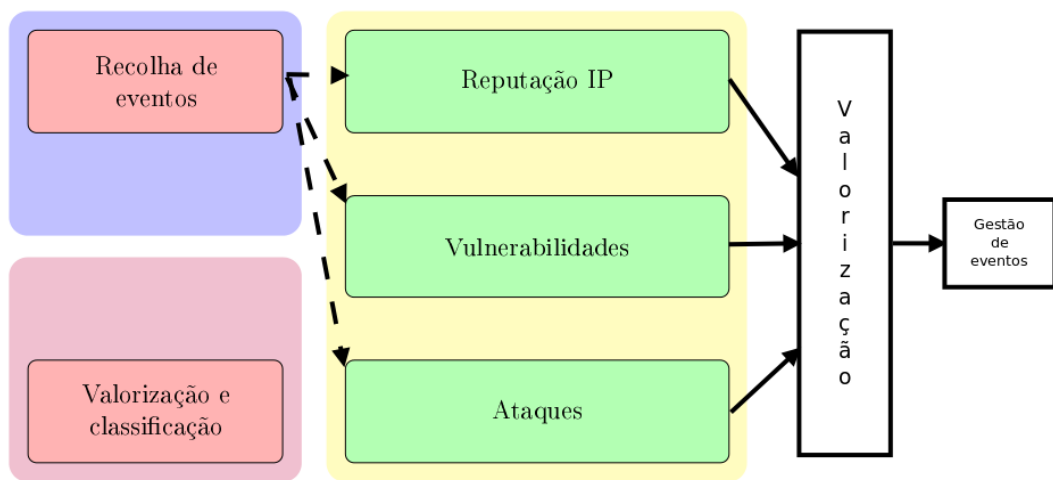


Figura 3.1: Arquitectura do sistema implementado

3.3.1.1 Módulo recolha de eventos

O módulo de recolha de eventos é responsável pela verificação e recolha de novos eventos. A cada dois segundos é verificada a existência de novos eventos, de forma a manter o sistema o mais actualizado possível. Dado que o *Snort* guarda toda a informação numa base de dados criada para o efeito, a verificação e recolha dos eventos é efectuada através de consultas a essa base de dados.

3.3.1.2 Módulo reputação IP

O módulo de Reputação IP é responsável por verificar se o endereço de origem associado a um evento, é considerado malicioso perante uma lista de endereços conotados como tal.

3.3.1.3 Módulo vulnerabilidades

No módulo de vulnerabilidades encontram-se todas as operações lógicas relativas à descoberta e associação de vulnerabilidades. Este módulo tem 3 funções principais distintas: verificar as vulnerabilidades presentes no sistema a monitorizar, verificar as vulnerabilidades associadas a cada evento e por fim efectuar o *match* entre elas.

3.3.1.4 Módulo ataques

No módulo de ataques estão implementadas todas as funcionalidades relativas à análise e associação de ataques, que um evento possa ter associado. É neste módulo que são efectuadas as operações para a criação da lista de ataques e a análise do tipo de *malware* associado a um evento, com o auxílio da lista de ataques CAPEC.

3.3.1.5 Módulo valorização e classificação

O módulo de valorização e classificação é responsável pelo calculo da criticidade dos eventos de segurança analisados. É neste módulo que estão implementadas as funcionalidades relativas à valorização e priorização dos eventos gerados e analisados.

3.3.2 Implementação do sistema

Conhecida a arquitectura do sistema, segue-se a descrição da sua implementação tendo em conta os requisitos especificados. Nesta secção serão descritas as opções tomadas quanto à implementação do sistema.

O sistema desenvolvido é composto por um conjunto de *scripts* escritas em *Python*. *Python* é uma linguagem de programação interpretada de alto nível e orientada a objectos. É uma linguagem que se caracteriza pela sua funcionalidade, rapidez de execução, simplicidade da sua sintaxe e legibilidade do código. A escolha desta linguagem de programação deve-se principalmente ao facto de ser uma linguagem orientada à prototipagem e à rapidez de execução do código. A rapidez de execução era uma preocupação devido ao facto do sistema desenvolvido consistir numa *framework* para gestão e classificação de eventos de segurança, onde a utilização de sistemas "leves" e "rápidos" é um aspecto com algum peso. (Dubois 2007)

Cada *script* do sistema consiste numa classe, estando implementado em cada

uma delas os seus respectivos métodos e atributos. Na figura 3.2 é possível visualizar o diagrama de classes do sistema.

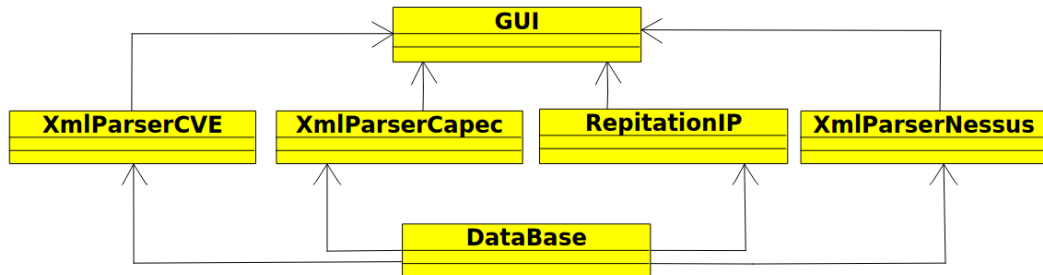


Figura 3.2: Diagrama de classes

3.3.2.1 Classes da aplicação

Classe DataBase.py

Na classe **DataBase.py** (figura A.1) são implementadas todas as funções que interagem com a base de dados do *Snort*. Através da biblioteca *mysqldb* é efectuada uma ligação à base de dados *MySql* onde o *Snort* armazena toda a informação relativa aos eventos gerados.

Classe XmlParserCVE.py

Na classe **XmlParserCVE.py** (figura A.2) estão implementados os métodos que possibilitam recolher informação proveniente da lista de vulnerabilidades (CVE). Esta lista consiste num ficheiro escrito em XML. Dado que o XML é um formato de dados hierárquico, foi necessário o uso da biblioteca *ElementTree* de forma a fazer a leitura dos dados, colocando-os numa estrutura em árvore.

Classe XmlParserCapec.py

Na classe **XmlParserCapec.py** (figura A.3) estão implementados os métodos que possibilitam recolher informação proveniente da lista de ataques (CAPEC), que tal como a lista de vulnerabilidade, é um ficheiro XML, sendo usada a biblioteca *ElementTree*.

Classe `XmlParserNessus.py`

Na classe `XmlParserNessus.py` (figura A.4) foram implementados os métodos que possibilitam efectuar a leitura do ficheiro gerado pelo *NESSUS*. Dado que este relatório também é escrito em XML, a biblioteca *ElementTree* foi utilizada tal como nas classes *scripts* `XmlParserCapec.py` e `XmlParserCVE.py`.

Classe `ReputationIP.py`

A classe `ReputationIP.py` (figura A.5) tem como função verificar se um dado IP consta na lista de IPs que estão identificados como maliciosos.

Classe `Gui.py`

A classe `Gui.py` (figura A.6) é a classe principal do sistema, consistindo também numa interface gráfica, de forma ao sistema interagir com o utilizador apresentando toda a informação. Esta *script* utiliza uma biblioteca denominada de *PyGtk* que possibilita a criação de interfaces gráficas.

3.3.2.2 Algoritmos do modelo desenvolvido

Algoritmos de verificação e recolha de eventos

Os algoritmos de recolha e verificação de eventos (algoritmos 1 e 2) têm como função verificar periodicamente a existência de novos eventos, bem como recolher esses mesmos eventos. De forma a saber quais são os novos eventos, o sistema guarda a data do ultimo evento que foi recolhido para poder comparar na próxima verificação, ocorrendo estas em cada dois segundos. Quando existirem novos eventos este recolhe e actualiza a lista de eventos.

Algoritmo 1: Atualização da lista de eventos

Data: dateTime = 0000-00-00 00:00:00

Result:

```
1 initialization;
2 while True do
3     system.sleep(2);
4     if getNewEvents().isEmpty == False then
5         dateTime = getNewEvents().eventTime;
6         updated();
7     end
8 end
```

Algoritmo 2: Recolha de novos eventos

Data: dbConnect, eventList

Result: Lista de eventos

```
1 initialization;
2 eventList = NULL;
3 while dbConnect.select() != NULL do
4     event = dbConnect.select();
5     eventList.append(event);
6 end
7 return eventList;
```

Algoritmo Criticidade

No sistema implementado, como já anteriormente descrito, a criticidade (algoritmo 3) de um evento é dada pela perigosidade do evento, se o endereço se encontra na lista de reputação IP e pelo tipo de ataque com maior peso.

Algoritmo 3: Cálculo da criticidade

Data: eventList

Result:

```
1 initialization;
2 criticity = [ ];
3 for event in eventList do
4     criticity.append(getHazard());
5     criticity.append(getAttackType());
6     criticity.append(getPresentIpReputation());
7 end
8 return criticity;
```

A perigosidade de um evento (algoritmo 4) é calculada através de diversos parâmetros oriundos das vulnerabilidades do evento, dos ataques descritos na lista de ataques e de informação disponível pelo *scanning* de vulnerabilidades (relatório *Nessus*).

Algoritmo 4: Cálculo da perigosidade

Data: event
Result: hazard

- 1 initialization;
- 2 hazard = hazard + hazardVulnerabilities();
- 3 hazard = hazard + hazardAttacks();
- 4 hazard = hazard + hazardNessusInfo();
- 5 **return** hazard;

Relativamente às vulnerabilidades de cada evento é necessário verificar quais se encontram no sistema a monitorizar (algoritmo 5). Através do endereço do evento verifica-se quais as vulnerabilidades presente no relatório *Nessus* (relativas a esse endereço).

Algoritmo 5: Correspondência de vulnerabilidades

Data: event, nessusReport
Result: vulnerabilities

- 1 initialization;
- 2 vulnerabilities = [];
- 3 **for** *eVulnerability* **in** *event.vulnerabilities()* **do**
- 4 **for** *nVulnerability* **in** *nessusReport.vulnerabilities(event.dstIp)* **do**
- 5 **if** *eVulnerability* == *nVulnerability* **then**
- 6 vulnerabilities.append(*eVulnerability*);
- 7 **end**
- 8 **end**
- 9 **end**
- 10 **return** vulnerabilities;

Efectuada a correspondência de vulnerabilidades do evento e as presentes no relatório *Nessus*, é calcula a perigosidade relativa às vulnerabilidades relativas a essa correspondência. No algoritmo 6 é possível verificar essa processo, onde são utilizados os valores do *status* de cada vulnerabilidade e o ano em que esta foi reportada na lista de vulnerabilidades CVE.

Algoritmo 6: Cálculo da perigosidade relativa às vulnerabilidades

Data: event
Result: hazardVulnerabilities

```
1 initialization;  
2 hazardVulnerabilities = 0;  
3 for vulnerability in event.getVulnerabilitiesMatch() do  
4   | hazardVulnerabilities = hazardVulnerabilities +  
   | vulnerability.getStatusValue();  
5   | hazardVulnerabilities = hazardVulnerabilities +  
   | vulnerability.getOldValue();  
6 end  
7 return hazardVulnerabilities;
```

Para cada evento analisado é necessário identificar os ataques que este pode representar. Os ataques presentes na lista de ataques CAPEC e que a sua descrição contenha 60% de correspondência em relação a palavras chave contidas na assinatura do evento, são adicionados a lista de ataques. No algoritmo 7 é possível verificar como são identificados os ataques.

Algoritmo 7: Identificação de ataques

Data: event, capecList
Result: attacks

```
1 initialization;  
2 attacks = [ ];  
3 for attack in capecList.allAttacks() do  
4   | if matchStr(attack.description, event.signature)  $\geq 60\%$  then  
5   | | attacks.append(attack);  
6   | end  
7 end  
8 return attacks;
```

Para cada ataque descrito na lista são utilizados os seguintes valores para o cálculo da perigosidade: severidade, dificuldade de realização/concretização, nível de perícia do atacante e o impacto relativo às propriedades de confidencialidade, integridade e disponibilidade. No algoritmo 8 é possível verificar como é efectuado o cálculo da perigosidade relativa aos ataques.

Algoritmo 8: Cálculo da perigosidade relativa aos ataques

Data: event
Result: hazardAttacks

- 1 initialization;
- 2 hazardAttacks = 0;
- 3 **for** *attack* **in** *attacks.attackList* **do**
- 4 hazardAttacks = hazardAttacks + attack.severity;
- 5 hazardAttacks = hazardAttacks + attack.likelihoodofExploit;
- 6 hazardAttacks = hazardAttacks + attack.CIAimpact;
- 7 **end**
- 8 **return** hazardAttacks;

Relativamente ao relatório gerado pelo *scanner* de vulnerabilidades *Nessus*, as informações utilizadas para o cálculo da perigosidade são o factor de risco e a severidade, sendo estes relativos à porta de destino do evento. No algoritmo 9 é possível verificar todo o processo do calculo da perigosidade relativa a essa informação.

Algoritmo 9: Cálculo da perigosidade relativa ao relatório *Nessus*

Data: event, nessusReport
Result: hazardNessus

- 1 hazardNessus = nessusReport.severity(event.dstPort);
- 2 hazardNessus = hazardNessus.riskFactor(event.dstPort);
- 3 **return** hazardNessus;

Tipo de ataque

Em relação aos ataques que maior peso têm, em função do seu tipo, no algoritmo 10 é possível verificar como este é encontrado na lista de ataques relativa a cada evento.

Algoritmo 10: Descoberta tipo de ataque

Data: event, parameters
Result: attackType

```
1 list = parameters.sortbyPercent();
2 for par in list.getName() do
3   | if par in event.attackList then
4   |   | for attack.type() in event.attackList do
5   |   |   | if attack.tipe() == par then
6   |   |   |   | return attack.tipe()
7   |   |   | end
8   |   | end
9   | end
10 end
```

Reputação IP

Relativamente a reputação IP, é verificado se o endereço de origem do evento em análise se encontra na lista *Reputation IP* fornecida pela *AlienVault*. No algoritmo (algoritmo 11) é possível verificar essa operação.

Algoritmo 11: Reputação IP

Data: event, reputationList
Result: reputation

```
1 if event.srcIP in reputationList then
2   | return True;
3 else
4   | return False;
5 end
```

Capítulo 4

Testes e avaliações

Neste capítulo serão apresentados os testes e resultados efectuados em laboratório, de forma a efectuar uma validação da solução proposta.

Inicialmente é descrito o ambiente de teste bem como todos os componentes envolvidos, seguido da descrição e explicação dos testes realizados, detalhando os diferentes tipos e qual a sua finalidade para a validação do sistema desenvolvido.

Por último são apresentados todos os resultados da experiência prática realizada e as conclusões retiradas desta.

4.1 Ambiente de teste

Para a realização dos testes ao sistema, foi criado um "laboratório de segurança" que permitisse criar os vários cenários de ataque, bem como simular os diferentes contextos em que estes possam estar inseridos. Este laboratório consistiu numa rede virtual de cinco máquinas com características diferentes, para que os testes fossem mais abrangente possível.

Um dos pressupostos é que os testes teriam de ser realizados num ambiente o mais controlado possível, para que os resultados não fossem deturpados. Para isso foi decidido isolar o ambiente de teste de tudo o que poderia inicialmente afectar os resultados obtidos nos testes. Desta forma a rede virtual onde os testes foram efectuados estava fisicamente isolada de qualquer outra rede, para que existi-se controlo "total" do tráfego que circulava na rede virtual de teste.

O método utilizado para a realização dos testes foi a geração de tráfego malicioso na rede, para que fossem gerados eventos e desta forma verificar a resposta do sistema desenvolvido. Para isso foi utilizado a ferramenta de segurança *Metasploit*¹, que consiste numa ferramenta para a realização de testes de penetração. Esta ferramenta permite verificar o estado da segurança de sistemas computacionais, explorando vulnerabilidades e falhas de segurança de forma a gerar tráfego malicioso, invadindo esses sistemas. Esta ferramenta contém diversos *exploits* para esse efeito, permitindo a adição de mais, que consistem em *Scripts* criadas para explorar e tirar partido de fragilidades nos diversos sistemas.

De forma a facilitar os testes foi utilizado o *Armitage*² que consiste numa *framework* para a utilização do *Metasploit*. O *Armitage* permite a utilização do *Metasploit* através de uma interface gráfica amigável, onde é possível visualizar os *hosts* em teste, para além de que inclui as ferramentas necessárias para *scanning*, como por exemplo o *Nmap*.³

A rede utilizada para os testes consistiu em quatro *hosts* virtualizados que serviram de vítimas e uma outra invasora. Dois dos *hosts* consistiam em sistemas *Windows* e as restantes *Linux*. Tanto nos sistemas *Windows* como *Linux* existia uma distribuição dedicada para servidores e uma outra para dedicada ao uso comum. o *host* que serviu de invasor era composto por um sistema operativo, baseado em *Linux*, desenvolvido e criado para este propósito: testes de penetração e segurança.

Segue uma lista de cada máquina virtual, bem como um pequena descrição de cada uma:

- **Ubuntu 13.04** - Distribuição Linux de Código aberto.
- **CentOS 6** - Distribuição Linux de classe *Enterprise* para servidores.
- **Microsoft Windows xp Service Pack 2** - Sistema Operativo proprietário (Microsoft).
- **Microsoft Windows Server 2008 Release 2** - Sistema Operativo proprietário (Microsoft) de Servidores.

¹<http://www.metasploit.com/>

visitado em Setembro de 2013

²<http://www.fastandeasyhacking.com/>

visitado em Setembro de 2013

³Ferramenta que possibilita a descobertas de *hosts* de uma rede, bem como informação relativa aos mesmos como *port scanning* e detecção do sistema operativo.

<http://nmap.org/>

visitado em Setembro de 2013

- **BackTrack 5 Release 3** - Sistema Operativo Linux, que agrega um vasto conjunto de ferramentas para a realização de testes de penetração.

Cada *host* foi configurado com alguns serviços, de forma a expandir o número de diferentes tipos de ataque, bem como o explorar diferentes vulnerabilidades. Também foi desligado qualquer tipo de *software* de segurança que viesse por defeito no sistema operativo, como por exemplo: a *firewall*. Na tabela 4.1 é possível visualizar de forma resumida quais os serviços relativos a cada *host*.

| | Mysql Server | Apache HTTP Server | FileZilla FTP Server | Tomcat Web Server |
|-----------------------|-------------------------|-------------------------------|---------------------------------|------------------------------|
| Ubuntu 13.04 | X | X | | |
| CentOS 6 | X | X | | |
| Wind. XP sp2 | X | X | X | X |
| Wind. Ser.2008 | X | X | X | X |

Tabela 4.1: Serviços instalados nos diferentes *hosts*

O *host* utilizado para gerar tráfego malicioso consistiu como já referido numa distribuição *Linux*, denominada de *Backtrack*. Por defeito esta distribuição *Linux* contem o *Metasploit* e *Armitage*. No *host Ubuntu 13.04* foi instalado o IDS *Snort*, sendo neste *host* que foram recolhidos os eventos gerados e testados pelo sistema implementado.

4.2 Testes e Resultados

Para a realização dos testes foram seleccionados alguns *exploits* para explorar vulnerabilidades nos sistemas acima descritos, de forma a criar tráfego malicioso na rede e consequentemente a geração de alertas de segurança por parte do IDS usado. Para isso foi efectuada uma pequena pesquisa para averiguar quais as vulnerabilidades típicas de cada sistema e que *exploits* existem para tirar partida das mesmas. Houve também um esforço para gerar diferentes tipos de ataques, de forma a diversificar os testes.

4.2.1 Tipos de ataques

As várias vulnerabilidades em cada *host* permitem que inúmeros tipos de ataques explorem essas diferentes fragilidades. Cada tipo de ataque tem as suas características, e estes quando executados têm consequências diferentes. O que caracteriza um ataque e o diferencia dos demais é o efeito e consequência que este pode vir a ter sobre os sistemas computacionais.

De forma a diversificar quantitativamente e qualitativamente os testes realizados, foram seleccionados os seguintes tipos de ataques:

- ***Network Reconnaissance*** - Consiste no reconhecimento parcial ou integral de uma rede de computadores, de forma a obter informação sobre os *hosts* que constituem a rede. Essa informação pode ser: conhecimento dos endereços da rede de forma a saber que *hosts* estão activos, o estado das portas do *host*, serviços activos, reconhecimento de sistemas operativos e que mecanismos de segurança implementados existem. Embora este tipo de ataque não tenha nenhuma consequência directa ou maliciosa, *Network Reconnaissance* é considerado um ataque devido ao facto de ser a primeira etapa a realizar quando se planeia a execução de qualquer tipo de intrusão. Contudo, este tipo de ataque é um caso particular em que não é explorada nenhuma vulnerabilidade.
- ***Data Structure Attacks*** - Consiste na manipulação e exploração de estruturas de dados do sistema, de forma a comprometer as mesmas, violando as suas propriedades de gestão. A ambiguidade na concepção das estruturas de dados, resulta em vulnerabilidades que permitem seu acesso indevido.
- ***Resource Manipulation*** - Consiste na manipulação e exploração de um ou mais recursos, ou algum atributo dos mesmos, com o intuito de alterar algum aspecto do estado desse recurso, de forma a afectar o comportamento de alguma aplicação ou serviço activo, pondo em causa até mesmo a integridade de informação.
- ***Probabilistic Techniques*** - Consiste em técnicas probabilísticas para explorar e ultrapassar os mecanismos de segurança do alvo. Este tipo de ataque baseia-se na "tentativa erro" de um conjunto de hipóteses, de baixa probabilidade matemática, de forma a conseguir identificar propriedades de segurança não existentes.
- ***Exploitation of Authentication*** - Consiste na exploração de vulnerabilidades em sistemas de autenticação, de forma a obter uma

identidade falsa com a qual o sistema interage. Com a autenticação no sistema o invasor adquire os privilégios da identidade ao qual este se faz passar.

Em relação *Network Reconnaissance* foi utilizado uma ferramenta chamada *Nmap*⁴, de forma a descobrir quais os *hosts* que constituem a rede, bem como informação relativa de cada *host*.

Quanto aos ataques do tipo *Data Structure Attacks*, foram utilizados os seguintes *exploits*:

- easyftp cwd fixret
- easyftp list fixret
- oracle9i xdb pass
- describe

Em relação aos ataques do tipo *Resource Manipulation*, os *exploits* utilizados foram os seguintes:

- smb ms08 067 netapi
- ms03 007 ntdll webdav
- awstats configdir exec
- base qry common

Quanto aos ataques classificados como *Probabilistic Techniques*, no ensaio laboratorial foi apenas utilizado um *exploit*:

- tftp-bruteforce

Por fim, relativamente aos ataques do tipo *Exploitation of Authentication*, foram utilizados os seguintes *exploits*:

- ms01 026 dbldecode

⁴*Scanner* que possibilita a descoberta de *hosts* de uma rede, bem como as suas portas e o sistema operativo, <http://nmap.org/> Visitado em Setembro de 2013

- MySQL remote root authentication bypass

Na tabela C.1 é possível visualizar quais os *exploits* utilizados em cada *host*, da rede do ensaio laboratorial.

4.2.2 Resultados obtidos

Conhecido o ambiente de teste, bem como os diferentes ataques utilizados para validação do modelo proposto, segue-se a apresentação dos resultados obtidos.

4.2.2.1 Teste 1 - Resultados obtidos

Inicialmente considerou-se que todos os tipos de ataques tinham uma relevância de 50% no sistema a monitorizar como é possível observar na figura B.1.

Na máquina utilizada para gerar tráfego malicioso na rede foi efectuado o *scan* à rede 192.168.178.0/24, de forma a encontrar todos os *hosts* que a constituem. Através do *scanner Nmap* executou-se o seguinte comando:

- NMAP -min-hostgroup 96 -sV -n -T4 -0 -F --version-light 192.168.178.0/24

O *scan* efectuado resultou na descoberta de quatro *hosts*, como é possível verificar na figura B.4, sendo estas as máquinas que serviram de "vítimas" neste ensaio laboratorial.

Dado a realização deste *scan* o IDS que se encontrava a monitorizar a rede, gerou cento e treze alertas. Na tabela 4.2 é possível visualizar quais os alertas gerados pelo IDS, a quantidade de cada um bem como a criticidade atribuída pelo sistema implementado.

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|--------------------------------------|------------|-------------|--------------|
| FINGER query | 2 | 32 | 230 |
| SCAN namp XMAS | 20 | 33 | 240 |
| ICMP PING undefined code | 20 | 70 | 984 |
| ICMP Echo Reply undefined code | 11 | 70 | 1056 |
| ICMP Echo Reply | 14 | 55 | 744 |
| ICMP Ping | 20 | 34 | 312 |
| ICMP Destination Port Unreachable | 12 | 70 | 1656 |
| ICMP Echo Reply | 14 | 42 | 744 |
| Total | 113 | | |

Tabela 4.2: Testes e resultados - *scanning*

Cada *host* da rede foi então sujeito a um conjunto de ataques. Em relação ao *host* que continha o sistema operativo *Windows XP Sp2*, na seguinte listagem é possível verificar o *exploit* usado e qual a tabela de resultados referente a este:

- smb ms08 067 netapi - Tabela C.2
- easyftp cwd fixret - Tabela C.3

- easyftp list fixret - Tabela C.4
- oracle9i xdb pass - Tabela C.5
- ms01 026 dbldecode - Tabela C.6
- ms03 007 ntdll webdav - Tabela C.7

Relativamente ao *host* com o sistema operativo *Windows Ser. 2008*, segue-se uma listagem com os *exploits* utilizados e as tabelas de resultados:

- smb ms08 067 netapi - Tabela C.8
- easyftp cwd fixret - Tabela C.9
- easyftp list fixret - Tabela C.10
- ms01 026 dbldecode - Tabela C.11
- ms 03 007 ntdll webdav - Tabela C.12

Em relação ao *host* com o sistema operativo *Ubuntu 13.04*, segue-se uma listagem com os *exploits* utilizados e as tabelas de resultados:

- awstats configdir exec - Tabela C.13
- base qry common - Tabela C.14
- describe - Tabela C.15
- tftp-bruteforce - Tabela C.16
- MySQL Remote Root Authentication Bypass - Tabela C.17

Relativamente ao *host* com o sistema operativo *CentOS 6*, segue-se uma listagem com os *exploits* utilizados e as tabelas de resultados:

- tftp bruteforce - Tabela C.18
- MySQL Remote Root Authentication Bypass - Tabela C.19

4.2.2.2 Teste 2 - Resultados obtidos

No segundo teste efectuado considerou-se que os tipos de ataques tinham uma relevância de 95% (figura B.2), de forma a que os valores da criticidade aumentassem em relação ao primeiro teste.

No teste realizado com os parâmetros a 95%, tal como no primeiro teste, foi efectuado um *scanning* à rede, utilizando exactamente o mesmo comando no *nmap* do primeiro teste. Na tabela D.1 é possível verificar o resultado deste *scanning*, sendo que os alertas gerados pelo IDS são exactamente os mesmos do primeiro teste, contudo a criticidade de cada evento aumentou.

Em relação ao *host* com o sistema operativo *Windows XP Sp2*, na seguinte listagem é possível verificar qual a tabela de resultados para cada *exploit*:

- smb ms08 067 netapi - Tabela D.2
- easyftp cwd fixret - Tabela D.3
- easyftp list fixret - Tabela D.4
- oracle9i xdb pass - Tabela D.5
- ms01 026 dbldecode - Tabela D.6
- ms03 007 ntdll webdav - Tabela D.7

Relativamente ao *host* com o sistema operativo *Windows Ser 2008*, na seguinte listagem é possível verificar qual a tabela de resultados para cada *exploit*:

- smb ms08 067 netapi - Tabela D.8
- easyftp cwd fixret - Tabela D.9
- easyftp list fixret - Tabela D.10
- ms01 026 dbldecode - Tabela D.11
- ms 03 007 ntdll webdav - Tabela D.12

Relativamente ao *host* com o sistema operativo *Ubuntu 13.04*, na seguinte listagem é possível verificar qual a tabela de resultados para cada *exploit*:

- awstats configdir exec - Tabela D.13
- base qry common - Tabela D.14
- describe - Tabela D.15
- tftp-bruteforce - Tabela D.16
- MySQL Remote Root Authentication Bypass - Tabela D.17

Relativamente ao *host* com o sistema operativo *CentOS 6*, na seguinte listagem é possível verificar qual a tabela de resultados para cada *exploit*:

- tftp bruteforce - Tabela D.18
- MySQL Remote Root Authentication Bypass - Tabela D.19

4.2.2.3 Teste 3 - Resultados obtidos

No terceiro e ultimo teste ao modelo desenvolvido, considerou-se que os tipos de ataque tinham uma relevância de apenas 20% (figura B.3), de forma a diversificar os testes e obtendo resultados mais conclusivos. Como se considerou que os ataques tinham uma relevância de 20%, era esperado que a criticidade dos eventos resultantes deste teste, fosse inferior comparativamente à criticidade dos mesmos eventos dos testes anteriores. Na tabela E.1 é possível visualizar os resultados relativo ao *scan* efectuado a rede.

Em relação ao *host* com o sistema operativo *Windows XP Sp2*, na seguinte listagem é possível verificar qual a tabela de resultados para cada *exploit*:

- smb ms08 067 netapi - Tabela E.2
- easyftp cwd fixret - Tabela E.3
- easyftp list fixret - Tabela E.4
- oracle9i xdb pass - Tabela E.5
- ms01 026 dbldecode - Tabela E.6
- ms03 007 ntdll webdav - Tabela E.7

Relativamente ao *host* com o sistema operativo *Windows Ser 2008*, na seguinte listagem é possível verificar qual a tabela de resultados para cada *exploit*:

- smb ms08 067 netapi - Tabela E.8
- easyftp cwd fixret - Tabela E.9
- easyftp list fixret - Tabela E.10
- ms01 026 dbldecode - Tabela E.11
- ms 03 007 ntdll webdav - Tabela E.12

Relativamente ao *host* com o sistema operativo *Ubuntu 13.04*, na seguinte listagem é possível verificar qual a tabela de resultados para cada *exploit*:

- awstats configdir exec - Tabela E.13
- base qry common - Tabela E.14
- describe - Tabela E.15
- tftp-bruteforce - Tabela E.16
- MySQL Remote Root Authentication Bypass - Tabela E.17

Relativamente ao *host* com o sistema operativo *CentOS 6*, na seguinte listagem é possível verificar qual a tabela de resultados para cada *exploit*:

- tftp bruteforce - Tabela E.18
- MySQL Remote Root Authentication Bypass - Tabela E.19

Capítulo 5

Conclusão e trabalho futuro

O presente capítulo conclui toda a descrição sobre o trabalho realizado ao longo deste projecto, sendo efectuada uma reflexão sobre os resultados obtidos em função dos objectivos inicialmente propostos. É também efectuada uma análise crítica de forma a abordar todos os objectivos e aspectos que não foram passíveis de ser alcançados em toda a sua plenitude. Por fim é descrito o trabalho a realizar posteriormente, de forma a melhorar e aperfeiçoar tanto o sistema implementado como o modelo desenvolvido.

5.1 Conclusões do projecto

Na presente dissertação foi apresentada uma proposta para um modelo capaz de classificar e valorizar eventos de segurança, de forma a ser plausível priorizar esses eventos em função da sua relevância, sendo esta a principal finalidade deste trabalho.

Inicialmente foi realizado um levantamento do estado da arte, onde é efectuada uma abordagem genérica sobre as redes de computadores, de forma a perceber as suas principais falhas e vulnerabilidades. Foi também efectuada uma abordagem teórica sobre segurança em redes de computadores, com o foco na detecção de intrusões, modelação de ataques cibernéticos e modelação de vulnerabilidades. Por isso foram listadas algumas ferramentas de segurança direccionadas para a detecção de intrusões. Todas estas abordagens tiveram como objectivo esclarecer e perceber que eventos de segurança devem ser valorizados e quais as fontes de informação utilizadas para essa valorização.

A segunda fase deste projecto consistiu no desenvolvimento e implementação do modelo proposto. Nesta fase começou-se por definir todos os requisitos do sistema bem como a descrição dos vários cenários e contextos, seguido do desenho e especificação do mesmo. No final foi efectuada a implementação do sistema. Nesta fase teve como principal objectivo responder à sub-questão de investigação qual a relação entre as diferentes fontes de informação utilizadas para a valorização dos eventos de segunda.

A terceira e última fase deste projecto, consistiu na validação do modelo proposto e implementado. Com a criação de um ambiente de teste, foram criadas situações de forma a recolher informação relativa à eficácia da valorização de eventos de segurança, retirando as devidas conclusões.

5.2 Análise crítica

Concluído o trabalho proposto e efectuando uma análise funcional do sistema desenvolvido, este tem algumas limitações no que diz respeito à eficácia e desempenho relativamente ao propósito para que foi desenvolvido.

Uma das limitações do sistema desenvolvido é a dependência do tipo de configuração do IDS *Snort*, pois este permite diferentes maneiras de registar os eventos. O sistema desenvolvido apenas tem a capacidade de recolher os eventos de segurança presentes numa base de dados *MySQL*, sendo necessário o IDS estar configurado para guardar a informação numa base de dados deste tipo.

O sistema desenvolvido recolhe informação proveniente de diversos ficheiros escritos em XML, tais com CVE e Capec, sendo que para aumentar a rapidez com que o sistema acede à informação contida nesses ficheiros, optou-se por carregar esses ficheiros em memória quando iniciado o sistema. Dado que estes ficheiros são de dimensão considerável, existe alguma latência quando iniciado o sistema. Para além do mais, quando iniciado o sistema é necessário estabelecer uma ligação ao servidor *MySQL* para aceder a base de dados de onde o *Snort* faz o registo dos eventos, o que resulta numa demora sempre que inicializado o sistema.

Uma das funcionalidades implementadas no sistema é a identificação de possíveis ataques que possam estar associados aos eventos de segurança. Como já anteriormente descrito, esta funcionalidade baseia-se numa relação entre palavras contidas na assinatura do evento com palavras da descrição de

cada ataque presente na lista de ataques CAPEC. Contudo esta abordagem apresenta uma probabilidade de erro considerável, podendo mesmo identificar um ataque erradamente.

Um dos parâmetros da fórmula da criticidade são as vulnerabilidades associadas a cada evento que se encontram presentes no sistema computacional a monitorizar. Esta associação está dependente da quantidade de informação, relativamente às vulnerabilidades, presente nas regras do *IDS*. Por exemplo, se existir um défice ou até mesmo nenhuma informação relativamente às vulnerabilidades de um evento, este será classificado com um valor de criticidade abaixo do esperado.

O sistema desenvolvido utiliza vários ficheiros com informação de contexto para valorizar e classificar os eventos. Estes ficheiros sofrem actualizações regulares por parte das entidades que os criaram. Uma das limitações do sistema é o facto de não ter nenhum mecanismo capaz de verificar a existência de actualizações e respectiva actualização de cada um. Já em relação ao ficheiro gerado pelo *Nessus*, de forma a manter a informação relativa às vulnerabilidades e fragilidades do sistema a monitorizar, é necessário efectuar novamente todo o processo para manter o ficheiro actualizado.

Em relação aos testes efectuados em laboratório para a validação do modelo proposto, algumas condições não foram devidamente testadas. Um dos parâmetro da criticidade era o facto do endereço de origem do evento constar na lista de reputação IP ou não, sendo que esta condição não foi testada. Dado este facto, os valores obtidos nos testes realizados são relativos eventos cujo o endereço de origem nunca consta na lista de reputação IP.

Ainda relativamente aos testes efectuados, existiu um défice na variedade do tipo de ataques gerados para a respectiva validação do modelo. Para além disso, o número de *hosts* que serviram de vítimas, era reduzido pelo que os resultados obtidos apenas são referentes aos sistemas operativos dos *hosts* em causa, bem como dos serviços e vulnerabilidades referentes aos mesmos.

5.3 Trabalho futuro

Após a conclusão deste projecto, com o cumprimento de todos os objectivos inicialmente propostos, existe um conjunto de aspectos que podem ser explorados de forma a acrescentar novas funcionalidades e aumentar a capacidade de eficácia de todo o sistema. Existem ainda algumas melhorias

ao trabalho desenvolvido que podem ser efectuadas, e que devido a limitações temporais não foram possíveis de serem realizadas. Dado isto, como trabalho futuro é proposto:

- Permitir a recolha e análise de eventos de segurança oriundos de outros IDS e/ou de outras tecnologias e sensores que gerem eventos relativos à segurança de sistemas computacionais.
- Criar e implementar um modulo capaz de correlacionar eventos de diversos sensores, de modo a ser plausível extrair mais informação relativa aos eventos e a diminuir os falsos positivos.
- Aumentar o número de fontes de informação que permitam valorizar eventos de segurança, como por exemplo: a lista de fraquezas CWE também disponibilizada pela *Mitre*.
- Incluir indicadores estatísticos e probabilísticos, de forma a aumentar a informação disponível ao gestor e/ou administrador de rede, como por exemplo: índice de dispersão e frequência dos eventos de segurança.
- Criar um modulo capaz de aumentar a informação sobre as vulnerabilidades associadas a cada evento, através de informação de contexto e fontes externas.
- Reestruturar o algoritmo de identificação do tipo de ataque de cada evento analisado, de forma a diminuir a probabilidade de erro.
- Automatizar o processo de actualização do ficheiro relativo à lista *IP Reputation*.

Bibliografia

Capec. <http://capec.mitre.org/>.

CVE. <http://cve.mitre.org/>.

M.A. Alhomidi en M.J. Reed, 2012. Attack graphs representations. In *Computer Science and Electronic Engineering Conference (CEEC), 2012 4th*, p. 83–88.

S. Ariyapperuma en C.J. Mitchell, 2007. Security vulnerabilities in DNS and DNSSEC. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, p. 335–342.

Sean Barnum, Cigital Inc, Amit Sethi, en Cigital Inc, 2007. Attack Patterns as a Knowledge Resource for Building Secure Software.

S. M. Bellovin, april 1989. Security problems in the TCP/IP protocol suite. *SIGCOMM Comput. Commun. Rev.*, 19-2 (1989), 32–48.

B. Bruhadeshwar, K. Kothapalli, M. Poornima, en M. Divya, 2009. Routing Protocol Security Using Symmetric Key Based Techniques. In *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, p. 193–200.

S.A. Camtepe en B. Yener, 2007. Modeling and detection of complex attacks. In *Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm 2007. Third International Conference on*, p. 234–243.

Michel Cukier en S. Panjwani, 2009. Prioritizing Vulnerability Remediation by Determining Attacker-Targeted Vulnerabilities. *Security Privacy, IEEE*, 7-1 (2009), 42–48.

- Paul F. Dubois, 2007. Guest Editor's Introduction: Python: Batteries Included. *Computing in Science Engineering*, 9-3 (2007), 7-9.
- Jung-Ho Eom, Young-Ju Han, Seon-Ho Park, en Tai-Myoung Chung, 2008. Active Cyber Attack Model for Network System's Vulnerability Assessment. In *Information Science and Security, 2008. ICISS. International Conference on*, p. 153-158.
- R. Gabriel, T. Hoppe, A. Pastwa, en S. Sowa, 2009. Analyzing Malware Log Data to Support Security Information and Event Management: Some Research Results. In *Advances in Databases, Knowledge, and Data Applications, 2009. DBKDA '09. First International Conference on*, p. 108-113.
- M. Garuba, Chunmei Liu, en D. Fraites, 2008. Intrusion Techniques: Comparative Study of Network Intrusion Detection Systems. In *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on*, p. 592-598.
- D. Gupta, P. S. Joshi, A. K. Bhattacharjee, en R. S. Mundada, 2012. IDS alerts classification using knowledge-based evaluation. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, p. 1-8.
- Wang Hao-yu, Zhu Xu, Cao Hui-zhi, Ji Chao-jun, en Ji Xiao-juan, 2010. The Security and Promotion Method of Transport Layer of TCP/IP Agreement. In *Information Technology and Computer Science (ITCS), 2010 Second International Conference on*, p. 513-517.
- Guofei Jiang, 2002. Multiple vulnerabilities in SNMP. *Computer*, 35-4 (2002), 2-4.
- Karen Kent en Murugiah P. Souppaya, 2006. SP 800-92. Guide to Computer Security Log Management. Technical report, Gaithersburg, MD, United States.
- James F. Kurose en Keith Ross, 2002. *Computer Networking: A Top-Down Approach Featuring the Internet*. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 2nd^e druk.
- Fuguo Li, 2012. Study on security and prevention strategies of computer network. In *Computer Science and Information Processing (CSIP), 2012 International Conference on*, p. 645-647.

-
- Yanyan Li en Keyu Jiang, 2012. Prospect for the Future Internet: A Study Based on TCP/IP Vulnerabilities. In *Computing, Measurement, Control and Sensor Network (CMCSN), 2012 International Conference on*, p. 52–55.
- Zhuowei Li, A. Das, en Jianying Zhou, 2005. Theoretical basis for intrusion detection. In *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, p. 184–192.
- J.M. Madrid, L.E. Munera, C.A. Montoya, J.D. Osorio, L.E. Cardenas, R. Bedoya, en C. Latorre, 2009. Functionality, reliability and adaptability improvements to the OSSIM information security console. In *Communications, 2009. LATINCOM '09. IEEE Latin-American Conference on*, p. 1–6.
- D. Miller, S. Harris, A. Harper, S. VanDyke, en C. Blask, 2010. *Security Information and Event Management (SIEM) Implementation*. McGraw-hill.
- Zhu Ning, Chen Xin-yuan, Zhang yong fu, en Xin Si-yuan, 2008. Design and Application of Penetration Attack Tree Model Oriented to Attack Resistance Test. In *Computer Science and Software Engineering, 2008 International Conference on*, deel 3, p. 622–626.
- Vern Paxson, 1998. Bro: a system for detecting network intruders in real-time. In *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7, SSYM'98*, p. 3–3, Berkeley, CA, USA. USENIX Association.
- Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, en Samir Chatterjee, december 2007. A Design Science Research Methodology for Information Systems Research. *J. Manage. Inf. Syst.*, 24-3 (2007), 45–77.
- Vladimir V. Riabov, 2007. *SMTP (Simple Mail Transfer Protocol)*, p. 388–406, in *Handbook of Computer Networks*. John Wiley & Sons, Inc.
- Robert Richardson, 2010/2011. CSI Computer Crime & Security Survey. (2010/2011).
- Martin Roesch, 1999a. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX conference on System administration, LISA '99*, p. 229–238, Berkeley, CA, USA. USENIX Association.
- Martin Roesch, 1999b. SNORT - Lightweight Intrusion Detection for Networks. (1999).

- Vineet Saini, Qiang Duan, en Vamsi Paruchuri, april 2008. Threat modeling using attack trees. *J. Comput. Sci. Coll.*, 23-4 (2008), 124–131.
- M. Sato, H. Yamaki, en H. Takakura, 2012. Unknown Attacks Detection Using Feature Extraction from Anomaly-Based IDS Alerts. In *Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on*, p. 273–277.
- M. Shimamura en K. Kono, 2006. Using Attack Information to Reduce False Positives in Network IDS. In *Computers and Communications, 2006. ISCC '06. Proceedings. 11th IEEE Symposium on*, p. 386–393.
- Murugiah. Souppaya, Karen. Kent, National Institute of Standards, en Technology (U.S.), 2006. *Guide to computer security log management [electronic resource] : recommendations of the National Institute of Standards and Technology / Murugiah Souppaya, Karen Kent*. U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology Gaithersburg, MD, draft.^e druk.
- M. Sourour, B. Adel, en A. Tarek, 2009. Environmental awareness intrusion detection and prevention system toward reducing false positives and false negatives. In *Computational Intelligence in Cyber Security, 2009. CICS '09. IEEE Symposium on*, p. 107–114.
- David Swift, 2006. *A Practical Application of SIM/SEM/SIEM Automating Threat Identification*.
- M. Takesue, 2011. E-mail Sender Identification through Trusted Local Deposit-Agents. In *Network-Based Information Systems (NBIS), 2011 14th International Conference on*, p. 84–91.
- Liu Xia, Feng Chao-sheng, Yuan Ding, en Wang Can, 2010. Design of secure FTP system. In *Communications, Circuits and Systems (ICCCAS), 2010 International Conference on*, p. 270–273.
- Sun Xiaoling, 2011. The study on computer network security and precaution. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, deel 3, p. 1695–1698.
- Chen Yan-ping, Liu Dong-liang, en Guo Rui, 2010. Security and precaution on computer network. In *Future Information Technology and Management Engineering (FITME), 2010 International Conference on*, deel 1, p. 5–7.

- Xin Yue, Wei Chen, en Yantao Wang, 2009. The research of firewall technology in computer network security. In *Computational Intelligence and Industrial Applications, 2009. PACIIA 2009. Asia-Pacific Conference on*, deel 2, p. 421–424.
- Bin Zeng, Lu Yao, en ZhiChen Chen, 2010. A network intrusion detection system with the snooping agents. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, deel 3, p. V3–232–V3–236.
- Chenfeng Vincent Zhou, Christopher Leckie, en Shanika Karunasekera, 2010. A survey of coordinated attacks and collaborative intrusion detection.
- A. Zúquete, 2010. *Segurança em Redes Informáticas, 3a Edição Atualizada e Aumentada*. FCA - Editora de Informática, Lda.

Apêndice **A**

Classes da aplicação

| DataBase |
|--|
| connector host user passWord dbName |
| connectDB() getEvents() getEvent() getVulnerabilityDataByCid() getVulnerabilityDataByName() getVulnerability() getNumberOcorrencesByEvent() refreshEvents() |

Figura A.1: Classe *DataBase*

| XmlParserCVE |
|--|
| rootCVE |
| parseFileToString() getVulnerability() getDescription() getStatus() setCveOld() setStatusValue() setVulnerabilityData() getVulnerabilityMatch |

Figura A.2: Classe *XmlParserCVE*

| XmlParserCapec |
|--|
| rootCapec |
| parseFileToString() getCategoryByID() getCategoryByName() getCategoryName() getDescription() getAttackPrerequisites() getResourcesRequired() getAllAttackPatterns() getAttackPatternByID() getAttackPatternByName() getNameAttackPattern() getAttackPatternDescription() getTypicalSeverity() getTypicalLikelihoodofExploit() getMethodsOfAttack() getAttackerSkills() getProbingTechniques() getIndicatorsWarningsOfAttack() getSolutionsAndMitigations() getMotivationConsequences() getInjectionVector() getPayload() getActivationZone() getPayloadActivationImpact() getRelevantSecurityRequirements() getRelatedSecurityPrinciples() getPurposes() getCIAImpact() getCVEList() |

Figura A.3: Classe *XmlParserCapec*

| XmlParserNessus |
|---|
| rootNessus |
| parseFileToString() getHosts() getReports() getSeverityAndRiskFactor() getCveList() setValue() |

Figura A.4: Classe *XmlParserNessus*

| ReputationIP |
|--|
| ipList |
| getAllIPs() getOnlyIps() getTypeValue() getIpReputationDictionary() |

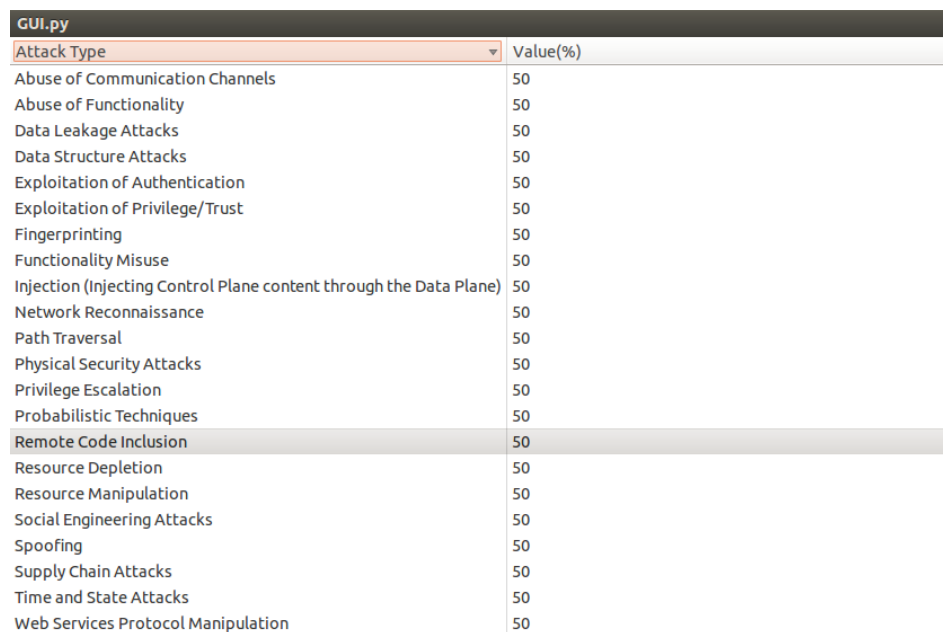
Figura A.5: Classe *ReputationIP*

| GUI |
|--|
| dbConnect rootCapec rootCVE rootNessus ipRep |
| getCriticity() getNessusInfo() getPresentIpReputation() showAttakSenario() showVulnerabilities() getAttackTypes() |

Figura A.6: Classe *GUI*

Apêndice B

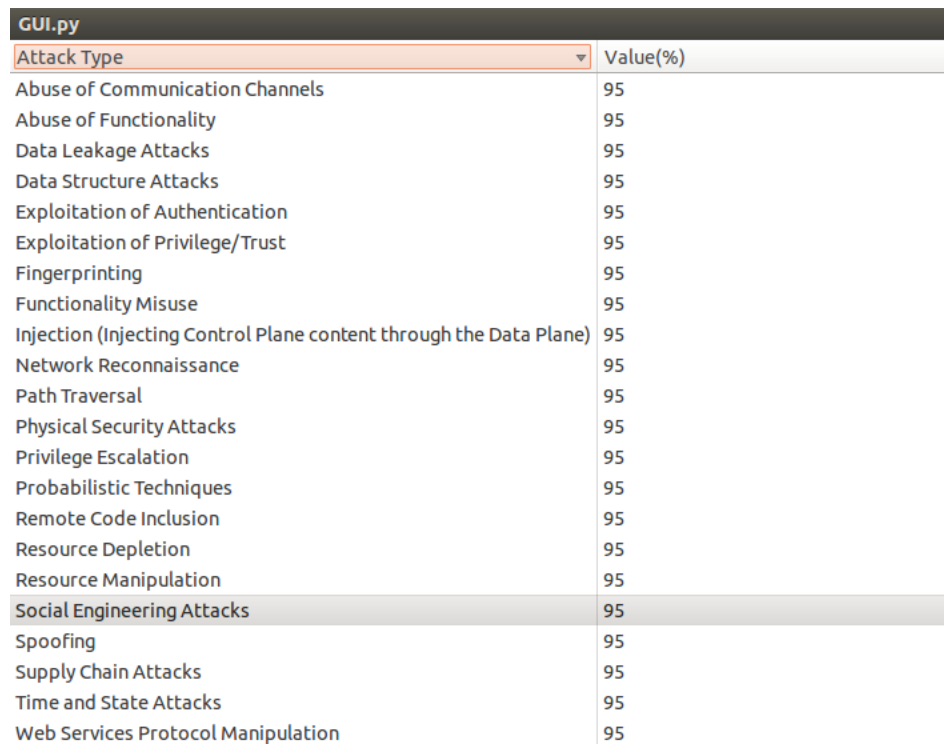
Imagens da aplicação



| Attack Type | Value(%) |
|--|----------|
| Abuse of Communication Channels | 50 |
| Abuse of Functionality | 50 |
| Data Leakage Attacks | 50 |
| Data Structure Attacks | 50 |
| Exploitation of Authentication | 50 |
| Exploitation of Privilege/Trust | 50 |
| Fingerprinting | 50 |
| Functionality Misuse | 50 |
| Injection (Injecting Control Plane content through the Data Plane) | 50 |
| Network Reconnaissance | 50 |
| Path Traversal | 50 |
| Physical Security Attacks | 50 |
| Privilege Escalation | 50 |
| Probabilistic Techniques | 50 |
| Remote Code Inclusion | 50 |
| Resource Depletion | 50 |
| Resource Manipulation | 50 |
| Social Engineering Attacks | 50 |
| Spoofing | 50 |
| Supply Chain Attacks | 50 |
| Time and State Attacks | 50 |
| Web Services Protocol Manipulation | 50 |

Figura B.1: Interface gráfica - preenchimento dos parâmetros relativos aos tipos de ataque(50%)

Modelo para definição de criticidade em eventos de segurança de redes de computadores



| Attack Type | Value(%) |
|--|----------|
| Abuse of Communication Channels | 95 |
| Abuse of Functionality | 95 |
| Data Leakage Attacks | 95 |
| Data Structure Attacks | 95 |
| Exploitation of Authentication | 95 |
| Exploitation of Privilege/Trust | 95 |
| Fingerprinting | 95 |
| Functionality Misuse | 95 |
| Injection (Injecting Control Plane content through the Data Plane) | 95 |
| Network Reconnaissance | 95 |
| Path Traversal | 95 |
| Physical Security Attacks | 95 |
| Privilege Escalation | 95 |
| Probabilistic Techniques | 95 |
| Remote Code Inclusion | 95 |
| Resource Depletion | 95 |
| Resource Manipulation | 95 |
| Social Engineering Attacks | 95 |
| Spoofing | 95 |
| Supply Chain Attacks | 95 |
| Time and State Attacks | 95 |
| Web Services Protocol Manipulation | 95 |

Figura B.2: Interface gráfica - preenchimento dos parâmetros relativos aos tipos de ataque(95%)

| Attack Type | Value(%) |
|--|----------|
| Abuse of Communication Channels | 20 |
| Abuse of Functionality | 20 |
| Data Leakage Attacks | 20 |
| Data Structure Attacks | 20 |
| Exploitation of Authentication | 20 |
| Exploitation of Privilege/Trust | 20 |
| Fingerprinting | 20 |
| Functionality Misuse | 20 |
| Injection (Injecting Control Plane content through the Data Plane) | 20 |
| Network Reconnaissance | 20 |
| Path Traversal | 20 |
| Physical Security Attacks | 20 |
| Privilege Escalation | 20 |
| Probabilistic Techniques | 20 |
| Remote Code Inclusion | 20 |
| Resource Depletion | 20 |
| Resource Manipulation | 20 |
| Social Engineering Attacks | 20 |
| Spoofing | 20 |
| Supply Chain Attacks | 20 |
| Time and State Attacks | 20 |
| Web Services Protocol Manipulation | 20 |

Figura B.3: Interface gráfica - preenchimento dos parâmetros relativos aos tipos de ataque(20%)

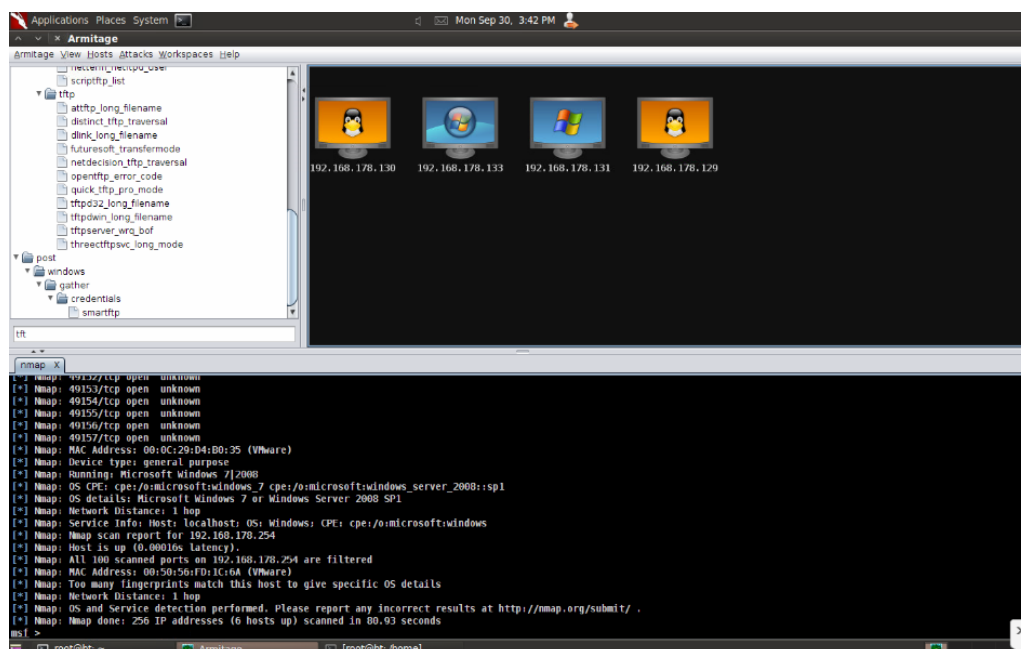


Figura B.4: Interface gráfica da ferramenta *Armitage*

Apêndice **C**

Tabelas de resultados

Modelo para definição de criticidade em eventos de segurança de redes de computadores

| | Ubuntu 13.04 | CentoOS 6 | Wind.XP sp2 | Wind.Ser. 2008 |
|------------------------------------|---------------------|------------------|--------------------|-----------------------|
| easyftp cwd fixret | | | X | X |
| easyftp list fixret | | | X | X |
| oracle9i xdb pass | | | X | |
| describe | X | | | |
| smb ms08 067 netapi | | | X | X |
| ms03 007 ntdll webdav | | | X | X |
| awstats configdir exec | X | | | |
| base qry common | X | | | |
| tftp bruteforce | X | X | | |
| ms01 026 dbldecode | | | X | X |
| MySQL rem. root auth. bypass | X | X | | |

Tabela C.1: Ataques realizados em cada *host*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|--------------------------------------|-------------------|--------------------|---------------------|
| NETBIOS SMB-DS IPC\$ share access | 1 | 36 | 291 |

Tabela C.2: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 49 | 1601 |
| FTP format string attempt | 1 | 70 | 2729 |
| FTP CWD overflow attempt | 1 | 50 | 1001 |
| INFO FTP BAG login | 1 | 48 | 936 |
| POLICY FTP anonymous login attempt | 1 | 42 | 737 |

Tabela C.3: Testes e resultados - *Windows XP Sp2*

Modelo para definição de criticidade em eventos de segurança de redes de computadores

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 49 | 1601 |
| FTP format string attempt | 1 | 70 | 2729 |
| FTP CWD overflow attempt | 1 | 50 | 1001 |
| INFO FTP BAG login | 1 | 48 | 936 |
| POLICY FTP anonymous login attempt | 1 | 42 | 737 |

Tabela C.4: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---|-------------------|--------------------|---------------------|
| WEB-MISC login authorization string | 1 | 70 | 3024 |
| COMMUNITY WEB-MISC mod_jrun overflow attempt | 1 | 22 | 168 |

Tabela C.5: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------|-------------------|--------------------|---------------------|
| WEB-ISS cmd.exe access | 2 | 29 | 165 |

Tabela C.6: Testes e resultados - *Windows XP Sp2*

APÊNDICE C. TABELAS DE RESULTADOS

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|----------------------------------|-------------------|--------------------|---------------------|
| WEB-MISC WebDav search access | 6 | 34 | 261 |

Tabela C.7: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|-----------------------|-------------------|--------------------|---------------------|
| NETBIOS SMB-DS IPC | 1 | 33 | 246 |

Tabela C.8: Testes e resultados - *Windows Server 2008*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 32 | 1601 |
| FTP format string attempt | 1 | 35 | 2273 |
| FTP CWD overflow attempt | 1 | 36 | 1001 |
| INFO FTP BAG login | 1 | 57 | 936 |
| POLICY FTP anonymous login attempt | 1 | 46 | 737 |

Tabela C.9: Testes e resultados - *Windows Server 2008*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 46 | 1601 |
| FTP RNFR attempt | 1 | 57 | 2273 |
| FTP MKD overflow attempt | 1 | 34 | 905 |
| INFO FTP BAG login | 1 | 35 | 936 |
| POLICY FTP anonymous login attempt | 1 | 32 | 737 |

Tabela C.10: Testes e resultados - *Windows Server 2008*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------|-------------------|--------------------|---------------------|
| WEB-ISS cmd.exe access | 2 | 22 | 165 |

Tabela C.11: Testes e resultados - *Windows Server 2008*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|----------------------------------|-------------------|--------------------|---------------------|
| WEB-MISC WebDav search access | 6 | 24 | 261 |

Tabela C.12: Testes e resultados - *Windows Server 2008*

APÊNDICE C. TABELAS DE RESULTADOS

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| WEB-ATTACKS perl execution attempt | 1 | 68 | 1608 |
| WEB-CGI awstats access | 1 | 27 | 264 |

Tabela C.13: Testes e resultados - *Ubuntu* 13.04

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---|-------------------|--------------------|---------------------|
| COMMUNITY WEB-MISC BASE base_query_commom.php remote file include | 1 | 22 | 96 |

Tabela C.14: Testes e resultados - *Ubuntu* 13.04

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|--------------------------------------|-------------------|--------------------|---------------------|
| WEB-MISC http directory traversal | 2 | 38 | 624 |

Tabela C.15: Testes e resultados - *Ubuntu* 13.04

Modelo para definição de criticidade em eventos de segurança de redes de computadores

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|------------------------------|-------------------|--------------------|---------------------|
| TFTP Get | 7 | 30 | 336 |
| ICMP Destination Unreachable | | | |
| Port Unreachable | 7 | 70 | 1656 |

Tabela C.16: Testes e resultados - *Ubuntu* 13.04

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|----------------------|-------------------|--------------------|---------------------|
| MYSQL root | | | |
| login attempt | 1 | 38 | 624 |

Tabela C.17: Testes e resultados - *Ubuntu* 13.04

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|------------------------------|-------------------|--------------------|---------------------|
| TFTP Get | 7 | 30 | 336 |
| ICMP Destination Unreachable | | | |
| Port Unreachable | 7 | 70 | 1656 |

Tabela C.18: Testes e resultados - *CentOS* 6

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|-----------------------------|-------------------|--------------------|---------------------|
| MYSQL root login attempt | 1 | 38 | 624 |

Tabela C.19: Testes e resultados - *CentOS 6*

Apêndice **D**

Tabelas de resultados

Modelo para definição de criticidade em eventos de segurança de redes de computadores

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|--------------------------------------|-------------------|--------------------|---------------------|
| FINGER query | 2 | 46 | 230 |
| SCAN namp XMAS | 20 | 45 | 240 |
| ICMP PING undefined code | 20 | 87 | 984 |
| ICMP Echo Reply undefined code | 11 | 89 | 1056 |
| ICMP Echo Reply | 14 | 73 | 744 |
| ICMP Ping | 20 | 52 | 312 |
| ICMP Destination Port Unreachable | 12 | 88 | 1656 |
| ICMP Echo Reply | 14 | 73 | 744 |
| Total | 113 | | |

Tabela D.1: Testes e resultados - *scanning*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|--------------------------------------|-------------------|--------------------|---------------------|
| NETBIOS SMB-DS IPC\$ share access | 1 | 46 | 291 |

Tabela D.2: Testes e resultados - *Windows XP Sp2*

APÊNDICE D. TABELAS DE RESULTADOS

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 67 | 1601 |
| FTP format string attempt | 1 | 88 | 2729 |
| FTP CWD overflow attempt | 1 | 68 | 1001 |
| INFO FTP BAG login | 1 | 66 | 936 |
| POLICY FTP anonymous login attempt | 1 | 60 | 737 |

Tabela D.3: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 67 | 1601 |
| FTP format string attempt | 1 | 88 | 2729 |
| FTP CWD overflow attempt | 1 | 68 | 1001 |
| INFO FTP BAG login | 1 | 66 | 936 |
| POLICY FTP anonymous login attempt | 1 | 60 | 737 |

Tabela D.4: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---|-------------------|--------------------|---------------------|
| WEB-MISC login authorization string | 1 | 88 | 3024 |
| COMMUNITY WEB-MISC mod_jrun overflow attempt | 1 | 40 | 168 |

Tabela D.5: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------|-------------------|--------------------|---------------------|
| WEB-ISS cmd.exe access | 2 | 40 | 165 |

Tabela D.6: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|----------------------------------|-------------------|--------------------|---------------------|
| WEB-MISC WebDav search access | 6 | 42 | 261 |

Tabela D.7: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|-----------------------|-------------------|--------------------|---------------------|
| NETBIOS SMB-DS IPC | 1 | 42 | 246 |

Tabela D.8: Testes e resultados - *Windows Server 2008*

APÊNDICE D. TABELAS DE RESULTADOS

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 64 | 1601 |
| FTP format string attempt | 1 | 42 | 2273 |
| FTP CWD overflow attempt | 1 | 54 | 1001 |
| INFO FTP BAG login | 1 | 65 | 936 |
| POLICY FTP anonymous login attempt | 1 | 50 | 737s |

Tabela D.9: Testes e resultados - *Windows Server 2008*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 64 | 1601 |
| FTP RNFR attempt | 1 | 42 | 2273 |
| FTP MKD overflow attempt | 1 | 54 | 905 |
| INFO FTP BAG login | 1 | 65 | 936 |
| POLICY FTP anonymous login attempt | 1 | 50 | 737 |

Tabela D.10: Testes e resultados - *Windows Server 2008*

Modelo para definição de criticidade em eventos de segurança de redes de computadores

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------|-------------------|--------------------|---------------------|
| WEB-ISS cmd.exe access | 2 | 40 | 165 |

Tabela D.11: Testes e resultados - *Windows Server 2008*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|----------------------------------|-------------------|--------------------|---------------------|
| WEB-MISC WebDav search access | 6 | 42 | 261 |

Tabela D.12: Testes e resultados - *Windows Server 2008*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| WEB-ATTACKS perl execution attempt | 1 | 88 | 1608 |
| WEB-CGI awstats access | 1 | 42 | 264 |

Tabela D.13: Testes e resultados - *Ubuntu 13.04*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---|-------------------|--------------------|---------------------|
| COMMUNITY WEB-MISC BASE base_query_commom.php remote file include | 1 | 39 | 96 |

Tabela D.14: Testes e resultados - *Ubuntu 13.04*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|--------------------------------------|-------------------|--------------------|---------------------|
| WEB-MISC http directory traversal | 2 | 48 | 624 |

Tabela D.15: Testes e resultados - *Ubuntu 13.04*

APÊNDICE D. TABELAS DE RESULTADOS

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|--|-------------------|--------------------|---------------------|
| TFTP Get | 7 | 43 | 336 |
| ICMP Destination Unreachable Port Unreachable | 7 | 85 | 1656 |

Tabela D.16: Testes e resultados - *Ubuntu* 13.04

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|-----------------------------|-------------------|--------------------|---------------------|
| MYSQL root login attempt | 1 | 48 | 624 |

Tabela D.17: Testes e resultados - *Ubuntu* 13.04

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|--|-------------------|--------------------|---------------------|
| TFTP Get | 7 | 43 | 336 |
| ICMP Destination Unreachable Port Unreachable | 7 | 85 | 1656 |

Tabela D.18: Testes e resultados - *CentOS* 6

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|-----------------------------|-------------------|--------------------|---------------------|
| MYSQL root login attempt | 1 | 48 | 624 |

Tabela D.19: Testes e resultados - *CentOS 6*

Apêndice **E**

Tabelas de resultados

Modelo para definição de criticidade em eventos de segurança de redes de computadores

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|--------------------------------------|-------------------|--------------------|---------------------|
| FINGER query | 2 | 19 | 230 |
| SCAN namp XMAS | 20 | 21 | 240 |
| ICMP PING undefined code | 20 | 58 | 984 |
| ICMP Echo Reply undefined code | 11 | 58 | 1056 |
| ICMP Echo Reply | 14 | 43 | 744 |
| ICMP Ping | 20 | 22 | 312 |
| ICMP Destination Port Unreachable | 12 | 58 | 1656 |
| ICMP Echo Reply | 14 | 30 | 744 |
| Total | 113 | | |

Tabela E.1: Testes e resultados - *scanning*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|--------------------------------------|-------------------|--------------------|---------------------|
| NETBIOS SMB-DS IPC\$ share access | 1 | 16 | 291 |

Tabela E.2: Testes e resultados - *Windows XP Sp2*

APÊNDICE E. TABELAS DE RESULTADOS

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 31 | 1601 |
| FTP format string attempt | 1 | 56 | 2729 |
| FTP CWD overflow attempt | 1 | 38 | 1001 |
| INFO FTP BAG login | 1 | 36 | 936 |
| POLICY FTP anonymous login attempt | 1 | 30 | 737 |

Tabela E.3: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 31 | 1601 |
| FTP format string attempt | 1 | 56 | 2729 |
| FTP CWD overflow attempt | 1 | 38 | 1001 |
| INFO FTP BAG login | 1 | 36 | 936 |
| POLICY FTP anonymous login attempt | 1 | 30 | 737 |

Tabela E.4: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---|-------------------|--------------------|---------------------|
| WEB-MISC login authorization string | 1 | 58 | 3024 |
| COMMUNITY WEB-MISC mod_jrun overflow attempt | 1 | 10 | 168 |

Tabela E.5: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------|-------------------|--------------------|---------------------|
| WEB-ISS cmd.exe access | 2 | 10 | 165 |

Tabela E.6: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|----------------------------------|-------------------|--------------------|---------------------|
| WEB-MISC WebDav search access | 6 | 12 | 261 |

Tabela E.7: Testes e resultados - *Windows XP Sp2*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|-----------------------|-------------------|--------------------|---------------------|
| NETBIOS SMB-DS IPC | 1 | 12 | 246 |

Tabela E.8: Testes e resultados - *Windows Server 2008*

APÊNDICE E. TABELAS DE RESULTADOS

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 34 | 1601 |
| FTP format string attempt | 1 | 45 | 2273 |
| FTP CWD overflow attempt | 1 | 24 | 1001 |
| INFO FTP BAG login | 1 | 23 | 936 |
| POLICY FTP anonymous login attempt | 1 | 20 | 737 |

Tabela E.9: Testes e resultados - *Windows Server 2008*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| FTP command overflow attempt | 1 | 34 | 1601 |
| FTP RNFR attempt | 1 | 45 | 2273 |
| FTP MKD overflow attempt | 1 | 24 | 905 |
| INFO FTP BAG login | 1 | 23 | 936 |
| POLICY FTP anonymous login attempt | 1 | 20 | 737 |

Tabela E.10: Testes e resultados - *Windows Server 2008*

Modelo para definição de criticidade em eventos de segurança de redes de computadores

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------|-------------------|--------------------|---------------------|
| WEB-ISS cmd.exe access | 2 | 10 | 165 |

Tabela E.11: Testes e resultados - *Windows Server 2008*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|----------------------------------|-------------------|--------------------|---------------------|
| WEB-MISC WebDav search access | 6 | 12 | 261 |

Tabela E.12: Testes e resultados - *Windows Server 2008*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---------------------------------------|-------------------|--------------------|---------------------|
| WEB-ATTACKS perl execution attempt | 1 | 34 | 1608 |
| WEB-CGI awstats access | 1 | 12 | 264 |

Tabela E.13: Testes e resultados - *Ubuntu 13.04*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|---|-------------------|--------------------|---------------------|
| COMMUNITY WEB-MISC BASE base_query_commom.php remote file include | 1 | 9 | 96 |

Tabela E.14: Testes e resultados - *Ubuntu 13.04*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|--------------------------------------|-------------------|--------------------|---------------------|
| WEB-MISC http directory traversal | 2 | 18 | 624 |

Tabela E.15: Testes e resultados - *Ubuntu 13.04*

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|------------------------------|-------------------|--------------------|---------------------|
| TFTP Get | 7 | 13 | 336 |
| ICMP Destination Unreachable | | | |
| Port Unreachable | 7 | 35 | 1656 |

Tabela E.16: Testes e resultados - *Ubuntu* 13.04

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|----------------------|-------------------|--------------------|---------------------|
| MYSQL root | | | |
| login attempt | 1 | 18 | 624 |

Tabela E.17: Testes e resultados - *Ubuntu* 13.04

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|------------------------------|-------------------|--------------------|---------------------|
| TFTP Get | 7 | 13 | 336 |
| ICMP Destination Unreachable | | | |
| Port Unreachable | 7 | 35 | 1656 |

Tabela E.18: Testes e resultados - *CentOS* 6

Modelo para definição de criticidade em eventos de segurança de redes de computadores

| Alerta gerado | Quantidade | Criticidade | Perigosidade |
|-----------------------------|-------------------|--------------------|---------------------|
| MYSQL root login attempt | 1 | 18 | 624 |

Tabela E.19: Testes e resultados - *CentOS 6*