Universidade do Minho
Escola de Engenharia

Roberto Miguel Marques de Sousa

Modelo comportamental de ataques em redes informáticas

Outubro de 2014

Modelo comportamental de ataques em redes informáticas
Roberto Miguel Marques de Sousa
UMinho | 2014

Universidade do Minho
Escola de Engenharia

Roberto Miguel Marques de Sousa

Modelo comportamental de ataques em redes informáticas

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

Trabalho efetuado sob a orientação do
Professor Doutor Henrique Santos

Outubro de 2014

DECLARAÇÃO

Nome Roberto Miguel Marques de Sousa

Endereço electrónico: beto.sousa22@gmail.com Telefone: 919037599

Número do Bilhete de Identidade: 13386832

Título dissertação

Modelo Comportamental de ataques em redes informatica

Orientador:

Professor Doutor Henrique Santos

Ano de conclusão: 2014

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

Nos exemplares das teses de doutoramento ou de mestrado ou de outros trabalhos entregues para prestação de provas públicas nas universidades ou outros estabelecimentos de ensino, e dos quais é obrigatoriamente enviado um exemplar para depósito legal na Biblioteca Nacional e, pelo menos outro para a biblioteca da universidade respectiva, deve constar uma das seguintes declarações:

É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA TESE/TRABALHO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

Universidade do Minho, ____/____/_____

Assinatura: _____

Agradecimentos

Em qualquer jornada é fundamental a companhia e apoio dos que nos são mais próximos e esse apoio é mais sentido quanto mais árduo é o percurso.

Nos passados meses esse apoio foi essencial para conseguir completar a etapa mais importante e mais complexa no meu percurso académico.

Gostaria de começar por agradecer ao meu Orientador, o Professor Doutor Henrique Santos, por todo o tempo despendido em reuniões e acompanhamento. O seu papel foi inestimável no decorrer desta etapa, ajudando-me a aperfeiçoar as minhas competências e guiando-me nas minhas questões, sempre que necessário.

Queria também agradecer aos meus pais, pela paciência demonstrada nos últimos meses, mesmo quando o cansaço se acumulou e se sobrepôs ao bom humor.

Por me fazer rir quando mais precisava, apesar de não saber o quão importante é o sorriso de uma criança, e me ensinar que muitas vezes as soluções mais intuitivas são as mais eficazes, quero agradecer ao meu afilhado Tomás.

Por fim, gostaria de agradecer à minha namorada Vanessa Silva, que desde sempre foi a estrela que me conduziu e me apoiou, mesmo quando duvidei do meu sucesso. Sei que estes meses foram tão intrincados para ela como para mim e quero que saiba que o sucesso do meu percurso é em grande parte seu.

Obrigado.

Resumo

“All models are wrong, but some models are useful” - George E. P. Box

Na sociedade atual, a segurança nos sistemas de informação impõem-se como uma das temáticas de maior importância nas áreas relacionadas com as tecnologias de informação.

Existem várias ferramentas que visam apoiar os administradores de redes a detetar e prevenir ataques informáticos, sendo de inestimável valor os sistemas de deteção de intrusões. Infelizmente ainda não existe nenhum método capaz de mensurar a eficácia de um sistema de deteção de intrusões, pelo que ainda é desconhecido se o rácio de falsos positivos e falsos negativos é aceitável.

Um modelo de ataques para avaliação de sistemas de deteção de intrusões visa colmatar em parte esse problema. Graças a esse modelo, poderemos simular um ataque ao sistema e conhecer, em avanço, quais os eventos que o sistema de deteção de intrusões deveria despoletar.

Neste projeto, pretende-se criar um modelo que vise englobar o maior número de ataques possíveis e recolher os eventos gerados em cada um desses ataques a par com a criação de um *dataset* de tráfego malicioso.

Palavras-chave: *attack modeling, malware behavior, attack classification,* CAPEC*, network attack, intrusion detection system.*

Abstract

In current society, cyber security is one of the most important topics in the areas related to information technologies.

There are several tools with the purpose of assist network administrators to detect and prevent cyber-attacks and one of the most important tool is the Intrusion Detection System. Unfortunately, there still no method capable of measuring and improve the effectiveness of an intrusion detection system, and it is still very hard to find an acceptable ratio of false positives and false negatives.

A model of attacks for the evaluation of intrusion detection system is designed to help finding a solution for this problem. With a model off attacks, we can simulate an attack on a network and know in advance what events the intrusion detection system should trigger.

In this project, we will create a model that with the purpose of cover the greatest number of potential attacks and collect events and datasets generated in each one of these attacks along with the creation of a dataset of malicious traffic.

Keywords: *attack modeling, malware behavior, attack classification,* CAPEC*, network attack, intrusion detection system.*

Índice

Lista de Acrónimos

IP	Internet Protocol
IDS	Intrusion detection systems
DSRP	Design science research process
DDoS	Distributed Denial of Service
HIDS	Host-based intrusion detection system
NIDS	Network Intrusion Detection System
http	Hypertext Transfer Protocol
tcp	Transmission Control Protocol
ftp	File Transfer Protocol
DARPA	Defense Advanced Research Projects Agency
ACM	Attack Centric Model
TCM	Threat Centric Model
CWE	Common Weakness Enumeration
STRIDE	Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege
CAPEC	Common Attack Pattern Enumeration and Classification
OWASP	Open Web Application Security Project
DVWA	Damn Vulnerable Web App
CSRF	Cross-site request forgery

XSS	Cross-site scripting
DoS	Denial of Service
Udp	User Datagram Protocol
Tcp	Transmission Control Protocol
MAC	Media Access Control

Lista de Figuras

Lista de tabelas

Capítulo 1

1 Introdução

1.1 Motivação

Os sistemas de informação dominam o tecido socioeconómico global, sendo por isso essencial um funcionamento seguro e fluido dos sistemas de informação. A segurança nos sistemas de informação não é apenas essencial para as empresas da área, mas também e de forma cada vez mais relevante, para a infraestrutura socioeconómica. A qual vê diariamente a sua disponibilidade, integridade e confidencialidade ameaçada. A importância da segurança de informação ultrapassa barreiras fronteiriças e está a tornar-se uma nova estratégia de guerra. Os ataques informáticos são uma poderosa e ainda pouco explorada técnica ofensiva, podendo deixar um pais mergulhado no caos, como ocorreu em 2007 na Estónia.

Devido aos acontecimentos do último ano, relacionado com cyber espionagem envolvendo líderes de grande potências mundiais, os *media* alertaram a sociedade para a necessidade de proteção a nível dos sistemas de informação, quer pessoal quer a nível organizacional.

Por estes motivos e devido à rápida evolução da tecnologia, que tornaram possível a realização de ataques mais sofisticados e eficazes, é premente apostar em políticas, normas e modelos de prevenção. Estas políticas, normas e modelos visam detetar ataques por forma a impossibilitar ou dissuadir os atacantes.

Os atuais ataques demonstram que os criminosos estão, cada vez mais, intelectualmente instruídos e motivados para realizar ataques mais sofisticados, bem como em maior número [12]. O número de vulnerabilidades nos sistemas de informação aumentou exponencialmente na última década, mantendo o seu aumento relativamente constante nos últimos anos. Concomitantemente, aumentou o número

de ameaças, o que é deveras inquietante. Por outro lado, como pode ser observado na Figura 1.1, número de malware cresceu exponencialmente e tudo leva a querer, que este escalar se manterá no futuro próximo [3].

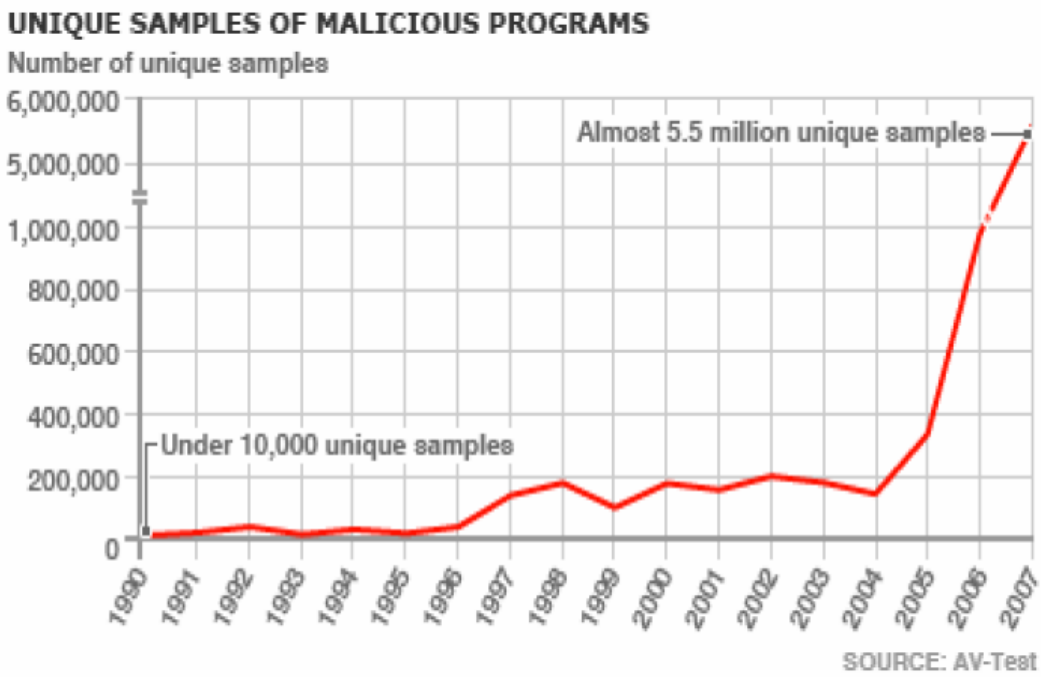


Figura 1.1 - Estatísticas do crescimento do malware [3]

Posto isto, é crescente a perceção do valor e necessidade da segurança de informação de dados, sendo notório o despertar da população para a necessidade da utilização de ferramentas para proteção das suas máquinas. Dentro dessas ferramentas, as mais comummente utilizadas são os antivírus e as *firewall*, sendo sem dúvida o antivírus o mais utilizado.

Um antivírus é um software de procura de padrões, normalmente sequencias de bytes já identificados, como malware. Dessa forma, sempre que é detetado malware, o utilizador é alertado atempadamente para a ameaça. Os antivírus recorrem a bases de conhecimento atualizadas frequentemente, como apoio à deteção de vírus mais recentes [27].

Uma *firewall*, em contraponto, tem como principal objetivo bloquear portas, endereços *Internet Protocol* (IP) ou serviços que possam fornecer acesso à máquina,

promovendo desta forma uma espécie de barreira entre a rede exterior e a rede interior. O bloqueio de serviços pela *firewall* é feito baseado em regras rígidas e pré-definidas, sendo estes bloqueios realizados indiscriminadamente, ou seja, sem nenhum tipo de análise de tráfego que seja proveniente de uma porta, endereço IP ou serviço bloqueado [17].

Para além das técnicas de proteção de informação mais utilizadas, os sistemas de deteção de intrusões (IDS) são uma ferramenta imprescindível e inestimável no âmbito da segurança de informação, permitindo por exemplo detetar de forma ágil e eficaz os ataques que resultam de um desvio do que pode ser considerado uma utilização normal do sistema. Os sistemas de deteção de intrusões baseiam-se em duas abordagens distintas, a deteção de assinaturas e a deteção de anomalias [9]. As duas abordagens referidas serão discutidas, com mais detalhe, no tópico 2.2.

Como referido, os sistemas de deteção de intrusões são uma ferramenta imprescindível e inestimável no âmbito da segurança de informação de rede, no entanto, a sua capacidade de deteção de intrusões ainda está longe da perfeição.

Um dos principais problemas que afeta estes sistemas, sendo premente a sua correção, é o elevado número de falsos positivos, os quais podem atingir 96% dos alertas gerados, como pode ser visto na Figura 1.2. Desta forma, uma elevada quantidade de tráfego é considerado malicioso mesmo não o sendo, o que torna a tarefa de análise de alertas extremamente complexa e morosa [29].

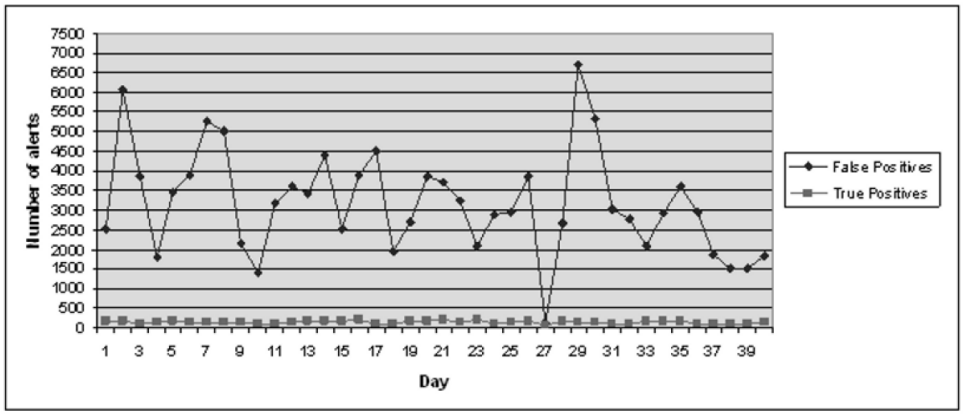


Figura 1.2 – Comparação entre alarmes falsos positivos e verdadeiros positivos [29]

Como tentativa de reduzir o número de falsos positivos, investigadores como Tjhai en Papadaki et al. 2008 [29] realizaram ajustes às regras de um sistema de deteção de intrusões com estes ajustes foi possível reduzir o número de falsos positivos de 96% para 87% no seu *dataset,* como pode ser visto na Figura 1.3 representando ainda um valor elevado [29].

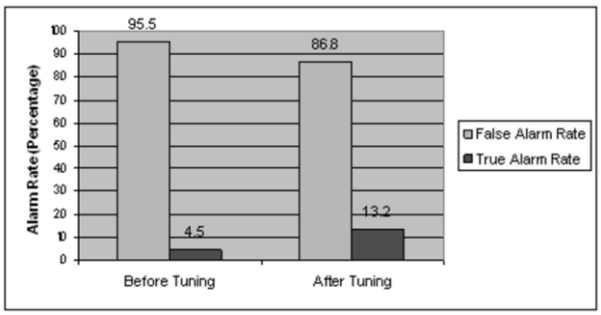


Figura 1.3 – Percentagem de falsos alarmes antes e depois do ajuste [29]

Desse modo, fazer ajustes para sistemas de deteção de intrusões, que se adequem a uma determinada rede, pode ser uma tarefa complexa e até ingrata, isto porque é imprescindível garantir o equilíbrio entre falsos negativos e falsos positivos, equilíbrio esse que é muito ténue [30].

Assim sendo a principal motivação para a presente dissertação é o desenvolvimento de um modelo de ataques robusto capaz de abranger um vasto número de ataques e passível de automação. Este modelo deverá conter um algoritmo de captação de *datasets,* para que com o apoio de um *software* adequado para injeção de tráfego na rede seja possível avaliar os sistemas de deteção de intrusões. Para esse efeito será essencial a aplicação de um vasto leque de ferramentas e conhecimento de técnicas de penetração.

1.2 Objetivos

O principal objetivo deste projeto foca-se no desenvolvimento de um modelo, o mais genérico possível, que consiga representar o decorrer de um ataque a sistemas de informação. O propósito deste modelo será determinar os eventos que um sistema de deteção de intrusões deve despoletar, assim como a recolha de tráfego de rede, para criação de um *dataset* apropriado, no decorrer de um ataque a redes informáticas.

Para o desenvolvimento do projeto será necessário estudar modelos já existentes, de modo a escolher o que mais se adequará ao propósito estabelecido. Após escolha devidamente justificada do modelo, será elaborada uma lista dos ataques na área de sistemas de informação, para ser testada em ambiente controlado.

Após criação do ambiente controlado dar-se-á início à simulação, com o propósito de recolher os *datasets,* assim como os eventos que serão usados como amostra nas fases de decisão do modelo, as quais serão determinadas posteriormente.

São objetivos deste projeto:

1. Escolher e adaptar, se necessário, um modelo de ataques existente;
2. Projetar e desenhar uma infraestrutura para testes em ambiente controlado;
3. Recolher eventos dos IDS e criar um *dataset* para avaliação de desempenho de sistemas de deteção de intrusões.

1.3 Método de investigação

Este projeto foi baseado num método de investigação para sistemas de informação [16], que divide uma investigação em seis fases distintas:

1. Identificação e motivação do problema;
2. Proposta de uma solução para o problema;
3. Desenvolvimento da solução proposta;
4. Demonstração;
5. Avaliação;
6. Escrita de resultados.

Este fluxo de etapas é facilmente identificável e representado de forma individual num modelo temporal, como pode ser visto na Figura 1.4.

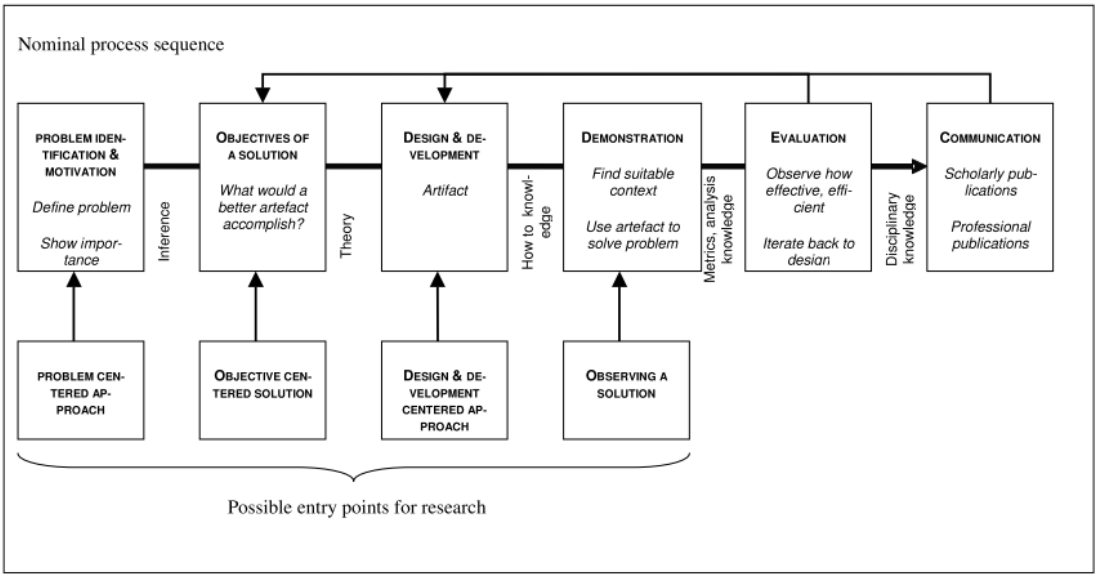


Figura 1.4 – Método de investigação proposto no Design science research process (DSRP) [16]

No caso concreto deste projeto, a pesquisa bibliográfica recolhida teve por base as seguintes palavras-chave: *attack modeling, malware behavior, attack classification,* CAPEC*, network attack, intrusion detection system.*

Recorrendo a apoio bibliográfico, foi elabora uma lista dos tipos dos modelos de ataques existentes e técnicas de modelação. Também como parte integrante do estudo, foram analisadas as diferentes bibliotecas de ataques existentes e as lacunas de cada uma.

Após definição detalhada dos objetivos, foi definido o modelo de ataques mais adequado e moldado por forma a cobrir da melhor forma os objetivos propostos inicialmente.

Para verificação dos resultados do projeto, bem como de todas as conclusões que dele derivam e demonstração da eficiência da solução proposta foi criado um sistema de teste, em ambientes virtuais controlados, recorrendo a sete máquinas virtuais.

Todos os resultados obtidos neste processo foram devidamente analisados e documentados. Baseado nas etapas previamente explicadas foi possível desenhar um diagrama de ação, que pode ser consultado no Apêndice B, com as principais questões a serem estudadas.

1.4 Estrutura da dissertação

No primeiro capítulo pretende-se enquadrar o tema do projeto, com uma breve introdução, chamando o leitor à atenção para a necessidade da criação de um modelo comportamental de ataques e explanando de forma sucinta qual o objetivo e motivação do projeto. Neste capítulo são ainda abordados quais os métodos de investigação utilizados e apresentada a estrutura a dissertação.

No segundo capítulo serão apresentados os conceitos fundamentais referentes à segurança de informação, descrevendo de forma breve a importância da segurança em redes na sociedade. Serão também apresentadas as principais características dos

diferentes tipos de sistemas de deteção de intrusões, discutindo o funcionamento, vantagens e desvantagens de cada um. Para além disso, este capítulo abriga um estudo detalhado sobre modelos de ataques em sistemas de informação, apresentando os tipos de modelos mais relevantes assim como as suas características fundamentais.

O terceiro capítulo estuda os vários modos de representação de modelos de ataques disponíveis, evidenciando algumas soluções já existentes apresentadas por outros investigadores. Posteriormente serão apresentadas as bibliotecas de ataques mais relevantes debatidas atualmente na comunidade científica e por fim, serão apresentadas as ferramentas utilizadas no decorrer do projeto.

No quarto capítulo serão descritos os requisitos do modelo de ataque, seguido do modelo proposto pelo autor da dissertação, com o propósito de sanar o problema inicialmente proposto. Também no mesmo capítulo será feita a escolha dos ataques mais relevantes a fazerem parte do estudo e posteriormente a ingressarem no *dataset.* Em seguida, descrever-se-á o ambiente de testes controlado que foi desenvolvido para o efeito. Será discutida a estratégia de recolha de eventos e tráfego de rede, bem como descritos detalhadamente dos ataques simulados e eventos recolhidos em cada um. Para finalizar o quarto capítulo, serão apresentados os eventos relevantes recolhidos durante a fase de testes assim como o *software* de injeção do tráfego malicioso na rede desenvolvido.

No quinto e último capítulo expor-se-ão as conclusões de todo o projeto. Será realizada uma análise crítica dos resultados obtidos e de todos os aspetos do projeto, salientando os aspetos positivos e aspetos a melhorar, bem como novas perspetivas que este projeto permitiu estabelecer.

Capítulo 2

2 Conceitos Fundamentais

O presente capítulo aborda os conceitos fundamentais referentes à segurança de informação, descrevendo de forma breve a importância da segurança em redes na sociedade. Este capítulo enquadra os conceitos teóricos relativos ao tema do projeto, descrevendo detalhadamente em que consiste um sistema de deteção de intrusões, assim como os principais desafios com que os sistemas de deteção de intrusões se deparam. Serão também descritos os seus métodos de recolha de dados assim como os dois tipos de sistemas de deteção de intrusões existentes.

2.1 Segurança de Informação na Sociedade

A crescente importância que os sistemas de informação e a internet têm para a nossa sociedade, para o nosso dia-a-dia e desenvolvimento económico gera naturalmente, preocupações relativas à sua estabilidade e segurança. Existem atualmente muitas tecnologias que, se bem aplicadas, contribuem para garantir a segurança e a confiança na utilização dos sistemas de informação. Porém, na área da segurança de informação, como aliás em todas as áreas onde a segurança é um fator relevante, a tecnologia só resolve os problemas se estiver integrada numa política de segurança bem definida, concebida de modo rigoroso e bem implementada. À escala mundial, a Internet só manterá o seu crescimento exponencial de utilização, se a rede for segura e ostentar elevado grau de qualidade nos serviços disponibilizados. Para a sociedade atual, quanto mais crucial é o uso da Internet, maior é o risco de esta ser alvo de tentativas de perturbação do seu funcionamento por quem o quer prejudicar, por exemplo, tentando causar falhas na rede, destruindo informação ou usando as redes como veículo de crimes.

Assim, como explicado acima, com a evolução da internet e com cada vez mais computadores ligados à rede, houve um aumento da necessidade de segurança da informação, pois grande parte dos negócios, economia mundial e até mesmo controlo de bens essenciais estão 100% dependentes da rede [17].

A segurança de informação tem como principais objetivo proteger três propriedades fundamentais, descritas abaixo e evidenciadas na Figura 2.1:

- **Confidencialidade**: Garante que dados ou determinado serviço estará apenas disponível aos utilizadores com privilégios para tal. Esta propriedade é imprescindível em empresas ou agências governamentais com dados sensíveis, como por exemplo patentes ou projetos de desenvolvimento de novos produtos.
- **Integridade**: Garante que os dados de um determinado sistema de informação não são alterados ou mesmo destruídos. Ataques à integridade têm como principal objetivo a destruição ou alteração de dados.
- **Disponibilidade**: Garante que os recursos necessários para o bom funcionamento do sistema estão sempre disponíveis. Os ataques à disponibilidade são uma grande ameaça aos sistemas de informação atuais. Estes visam consumir exageradamente recursos de um determinado sistema, desencadeando escassez de recursos para responder às necessidades reais do serviço. Uma das formas mais conhecidas de ataque à disponibilidade é o *distributed denial of service (DDoS)*.

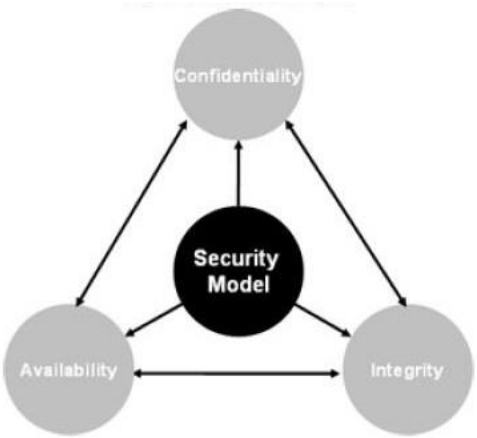


Figura 2.1 - Triângulo C.I.A.[1]

A segurança de informação é muito mais que proteger um simples componente. Especialistas defendem que a segurança de informação contém seis componentes essenciais, que necessitam de ser protegidos [17]:

- Software
- Hardware
- Dados
- Pessoal
- Procedimentos
- Redes

Um dos principais problema na segurança de informação é a falta de deteção atempada de um ataque. Os sistemas de deteção de intrusões visam, tal como o nome indica, detetar e agir de acordo com o estipulado pelo gestor de redes, podendo tomar medidas ou apenas alertar o gestor de rede para o sucedido. Esta ferramenta, essencial na segurança de informação, será explanada em termos do seu funcionamento no tópico seguinte.

[1] https://learningnetwork.cisco.com/servlet/JiveServlet/showImage/2-59912-10805/1.JPG

2.2 Sistemas de deteção de intrusões

Os sistemas de deteção de intrusões são sistemas que detetam ataques contra a rede ou máquinas na rede [18]. Como referido previamente, os sistemas de deteção de intrusões possuem duas abordagens de deteção de ataque, podendo utilizar apenas uma abordagens de deteção ou no caso de sistemas de deteção de intrusões mais complexos operar com as duas abordagens em simultâneo.

Por forma a compreender melhor a necessidade das duas abordagens iremos de seguida fazer uma breve analise sobre cada uma.

O sistema de deteção de intrusão que opera tendo por base as assinaturas, tenta detetar padrões de ataques conhecidos e assim prever tráfego malicioso.

Este tipo de sistema de deteção de intrusões, tem o grande inconveniente de necessitar de manter a sua base de dados de malware o mais atualizada possível. Com o crescimento do crime na web e de modo a dificultar a deteção de malware pelos sistemas de deteção de intrusões, o malware pode mudar as suas variantes. Estas mudanças permitem despistar assim os sistemas de deteção de intrusões que se baseiam na deteção por assinatura de malware [3]. O número de assinaturas assim como o seu respetivo tamanho tem vindo a aumentar drasticamente nos últimos anos. Para conter cada vez mais assinaturas de todos os ataques conhecidos, os sistemas de deteção de intrusões consomem muitos recursos, como armazenamento ou largura de banda, os quais são imprescindíveis para atualizar as assinaturas, como pode ser constatado nos gráficos apresentados na Figura 2.2 e Figura 2.3.

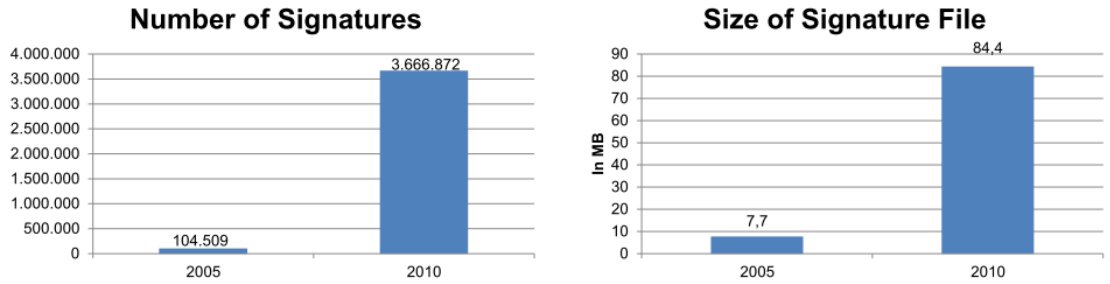


Figura 2.2 – Número de assinaturas e tamanho médio de cada em 2005 e 2010 [13]

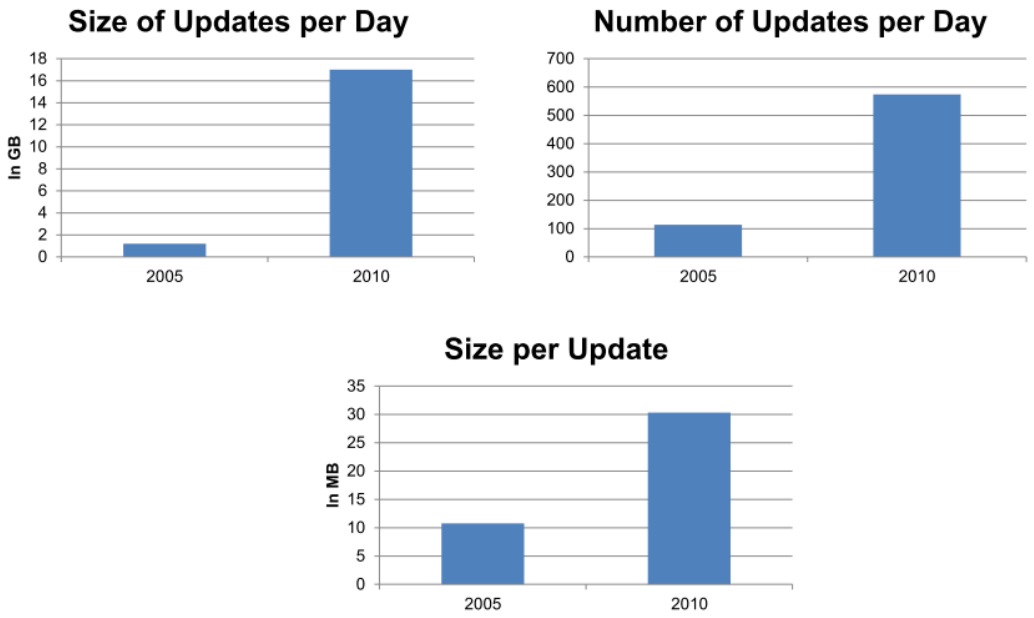


Figura 2.3 – Tráfego consumido na atualização de assinaturas em 2005 e 2010 [13]

Um outro tipo de sistema de deteção de intrusões é baseado em anomalias. Tendo conhecimento de um padrão normal de tráfego na rede e analisando esse mesmo tráfego, o sistema de deteção procura por algo dito anormal no tráfego, o que permite a deteção de ataques que ainda não sejam conhecidos pela assinatura [18].

Quanto à origem dos dados a analisar, existem dois tipos de sistemas de deteção de intrusões, o sistema de deteção de intrusões em rede (NIDS) e o sistema de deteção de intrusões no *host* (HIDS).

Os sistemas de deteção de intrusões na rede recolhem o tráfego da rede. Essa recolha é feita em modo promíscuo, ou seja, recolhe todo o tráfego existente na rede. Este tipo de monitorização funciona em tempo real, gerando alertas para o gestor de rede, bem como logs, os quais podem ser utilizados para uma investigação mais aprofundada dos ataques caso seja necessário.

De uma forma geral, os sistemas de intrusões baseados na rede analisam duas grandes componentes.

A primeira componente é conhecida por *protocol stack verification*. Esta componente do sistema de deteção de intrusões baseada na rede rastreia pacotes que não se encontram de acordo com as regras dos protocolos de rede. Por outro lado, o segundo componente, *application protocol verification* funciona a nível protocolar (http, ftp, tcp), rastreando violações de utilização de protocolos. Neste tipo de violação, a estrutura do protocolo está intacta, mas este é utilizado de forma errónea, sendo um exemplo muito comum o *syn overflow* [17].

Como principais vantagens dos sistemas de deteção de intrusões baseada na rede, são de destacar [17]:

- Passividade do sistema, podendo o sistema de deteção de intrusão baseado na rede atuar sem necessidade de instalação em todas as máquinas, tornando sua instalação mais transparente.
- Difícil alvo de ataques, pela resistente configuração, não sendo passível de deteção por parte do atacante.

Não obstante as vantagens referidas, o sistema de deteção de intrusões na rede apresenta limitações. Destaca-se a impossibilidade de analisar tráfego cifrado e a necessidade de configurações especiais para a recolha de todo o tráfego. Esta recolha nem sempre é possível devido à utilização de *switches*, que fazem as ligações ponto a ponto, obrigando o gestor de rede a configurar portas específicas de forma a possibilitar a recolha de todo o tráfego [17].

Por outro lado, os sistemas de deteção de intrusões baseados no *host* utilizam a recolha de logs tanto do tráfego como da máquina onde o sistema de deteção de intrusões está instalado, procurando detetar sinais de atividade maliciosa.

A principal função deste tipo de sistema de deteção de intrusões é verificar possíveis violações de integridade nos ficheiros do sistema, possíveis portas abertas indevidamente ou mais tipicamente tentativas falhadas de login, entre muitos outros sinais de anomalia.

Como nos sistemas de deteção de intrusões baseados no *host* implica necessariamente a instalação de um sistema por máquina, é possível enviar todos os eventos gerados para uma máquina central, centralizando o sistema e facilitando a monitorização de eventos por parte do gestor da rede.

Apesar dos sistemas de deteção de intrusões serem vitais para a deteção de tráfego malicioso na rede, tem limitações. De entre as limitações destacam-se como mais relevantes, o elevado número de falsos positivos, a falta de ferramentas e métricas disponíveis para avaliação e complexa comparação entre diferentes técnicas utilizadas em sistemas de deteção de intrusões [28].

O grande desafio atualmente nesta área, centra-se principalmente na redução do número de falsos positivos e na deteção novos ataques [29].

Infelizmente, os testes para avaliação de sistema de deteção de intrusões necessitam muito tempo para a sua realização, quer pela falta de tráfego malicioso, quer pela necessidade de avaliar a sua deteção e o número de eventos gerados pelos sistemas de deteção de intrusões [7]. De entre os esforços realizados para avaliar os sistemas de deteção de intrusões, destacam-se as técnicas que recorrem a *datasets*, já recolhidos, e as que utilizam uma rede em ambiente controlado, que permite realizar ataques pré-definidos [31].

Um dos *datasets* mais utilizados é fornecido pelo Defense Advanced Research Projects Agency (DARPA)[2] e remonta ao ano de 1999. É neste ponto que surge a necessidade da criação de um modelo comportamental de ataques em redes informáticas, genérico, abrangendo o maior número de ataques possíveis. Para além de ser genérico, o modelo comportamental elaborado deverá adicionalmente de conter um algoritmo de recolha de tráfego de rede, assim como dos eventos gerados pelos sistemas de deteção de intrusões, por forma a apoiar o avaliador do sistema de deteção de intrusões. O modelo elaborado deverá também ser passível de automação, permitindo a criação de uma ferramenta de injeção de tráfego malicioso na rede, gerando eventos para avaliação, minimizando o tempo despendido.

[2] http://www.darpa.mil/

2.3 Modelos de ataques em Sistemas de informação

Existem alguns tipos de modelos já propostos pela comunidade científica com especial foco no âmbito da análise de risco. Esses modelos centram-se em apenas dois tipos de modelos genéricos. O primeiro modelo genérico é o Attacker Centric Model (AC Model), que consiste num modelo que tenta descrever o comportamento do atacante na rede. Já o segundo modelo, Threat Centric Model (TC Model) é um modelo centrado nas possíveis ameaças e vulnerabilidades. Estes modelos serão explanados mais detalhadamente nos subtópicos seguintes.

2.3.1 Modelos centrados nos atacantes

O Modelo centrado no atacante baseia-se em duas grandes variáveis: o conhecimento que o atacante tem sobre a tecnologia alvo e a sua motivação para o ataque. Deste modo o modelo tenta prever o grau de perigosidade para a rede de acordo com esses dois fatores, exemplificando: um atacante categorizado como delinquente, com o nível de conhecimento de iniciante será uma ameaça menor, que um atacante com motivação categorizadas como criminosa e com conhecimento sólido. Apesar desta abordagem parecer aliciante à primeira vista, revela-se bastante complexa, pois torna-se difícil ou até mesmo impossível determinar a motivação e conhecimentos do atacante durante um ataque. Um outro ponto negativo desta técnica é descuidar, em parte, os recursos de rede focando-se principalmente no atacante [8].

A conjunção destes fatores pode representar diferentes níveis de perigo, como pode ser visto na Figura 2.4.

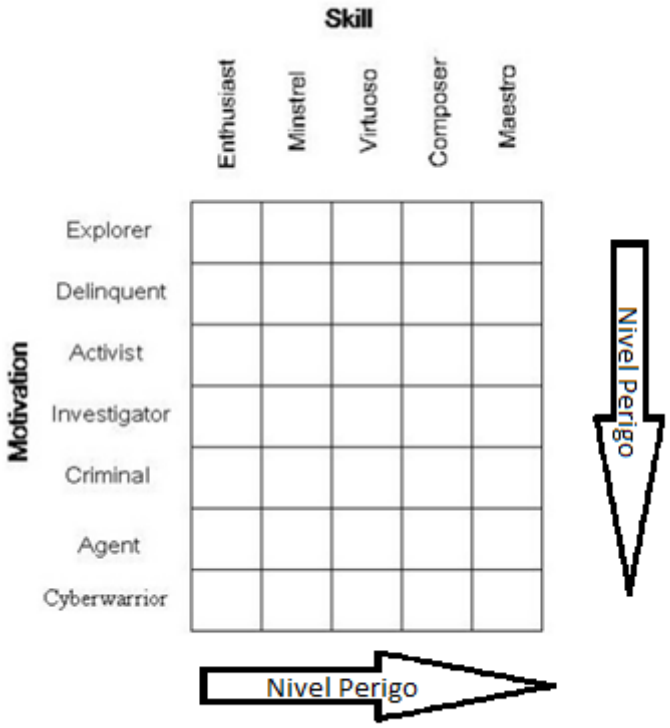


Figura 2.4 - Relação de entre a motivação e a aptidão ao ataque [7]

2.3.2 Modelo centrado nas ameaças ou recursos

Este tipo de modelo tem por objetivo desenhar um possível percurso que o atacante pode percorrer, de modo a poder atingir os seus objetivos. Para isso, é necessário ter conhecimento das possíveis vulnerabilidades da rede assim como do *software* utilizado para suportar a mesma.

De um modo geral, estes tipos de modelos são representados em árvore, sendo que em cada ramo da mesma estão representados recursos ou vulnerabilidades a serem exploradas pelos atacantes.

É importante neste tipo de modelo estar sempre atualizado relativamente às últimas vulnerabilidades publicadas, pois uma vulnerabilidade que tenha surgido recentemente num dos recursos disponíveis na rede, poderá criar toda uma nova falha de segurança, criando uma reação em cadeia.

Uma árvore de vulnerabilidades tem uma estrutura hierárquica, normalmente com ligações feitas com símbolos booleanos que visam representar por onde e como é que um sistema pode falhar.

Este tipo de modelação pode ser bastante eficaz na deteção e prevenção de vulnerabilidades antes dos próprios atacantes as detetarem, no entanto exige da parte do responsável pelo desenho da árvore um profundo conhecimento da rede e de todos os recursos que nela existem para determinar suas vulnerabilidades.

Pode-se concluir, que no desenho de modelos de ataques se torna difícil distinguir entre modelo de ameaças e modelo de ataques [8], pois ao representar as ameaças de um sistema, que deixam o mesmo vulnerável, temos também de ter conhecimento do ataque que irá explorar essa ameaça.

Capitulo 3

3 Especificações do Sistema

No presente capítulo serão apresentadas as técnicas de modelação de ataques em sistemas de informação assim como as soluções que existem atualmente na comunidade científica. Estes objetivam-se como mais relevantes para o decorrer desta investigação, a par das bibliotecas de ataques existentes.

Por fim será também apresentada uma descrição mais detalhada das ferramentas utilizadas na presente dissertação.

3.1 Modos de representação de modelos de ataques

Quando se torna premente o estudo de ataques, é fundamental representá-los de uma forma simples e percetível, possibilitando a uma terceira pessoa, externa à projeção do modelo de ataques a compreensão do mesmo. Essa representação pode ser feita de várias formas.

Nas secções seguintes, serão descritas as três técnicas de representação de modelos mais relevantes, utilizadas pela comunidade científica. Esta escolha foi feita baseada no estudo comparativo realizado por Mirembe en Muyeba et al. 2008 [8] cuja síntese pode ser observada na Figura 3.1.

Technique	Dynamic	Atomic Details	System Based	Attacker Focused	Semantics	Simplicity
Fault Trees	-	-	+	-	-	+
Attack Trees	-	-	-	+	-	+
Attack Suites	+	-	-	+	+	-
Attack Nets	+	+	-	+	+	-
Security Patterns	+	+	-	+	-	+

Figura 3.1 – Comparação de técnicas de representação de modelos [8]

Neste tópico serão explanados os modelos de representação, indicando as vantagens e desvantagens de cada modelo. Pretende-se deste modo analisar qual o modelo que melhor se enquadra nos objetivos do presente projeto.

3.1.1 Modelos Baseados em árvore

Os modelos baseados em árvore permitem uma estrutura hierárquica organizada, que flui da raiz da árvore para os seus extremos. Na raiz da árvore encontra-se o ponto inicial, que pode ser a posição inicial do atacante ou a primeira vulnerabilidade que pode/deve ser explorada, dependo do que estamos a representar na árvore (ataques, ameaças, etc).

Por uma questão de simplicidade e de coerência com o objetivo do projeto, deste ponto em diante, apenas se fará referência às árvores de ataques. É de referir, que estes pressupostos são também válidos para os pontos seguintes e para todos os tipos de modelos em árvore.

Nas folhas da árvore estão representados sub-objetivos necessários para a evolução do ataque. A transição entre folhas da árvore é feita sempre que uma ou mais

condições se verifiquem. Sempre que for necessário mais que uma condição para avançar para o sub-objetivo seguinte, esta deve ser representada com o símbolo lógico "*and*", que consiste na ligação de dois pontos, aos sub-objeticos, com uma linha curva.

Na Figura 3.2 pode ser observada uma árvore de ataques com vários sub-objectivos, onde no último ponto "*steal*" se encontra uma ligação lógica "*and*", que no exemplo em questão indica que para roubar uma palavra passe é necessário instalar um *sniffer* na vítima e conseguir receber os dados desse mesmo *sniffer*.

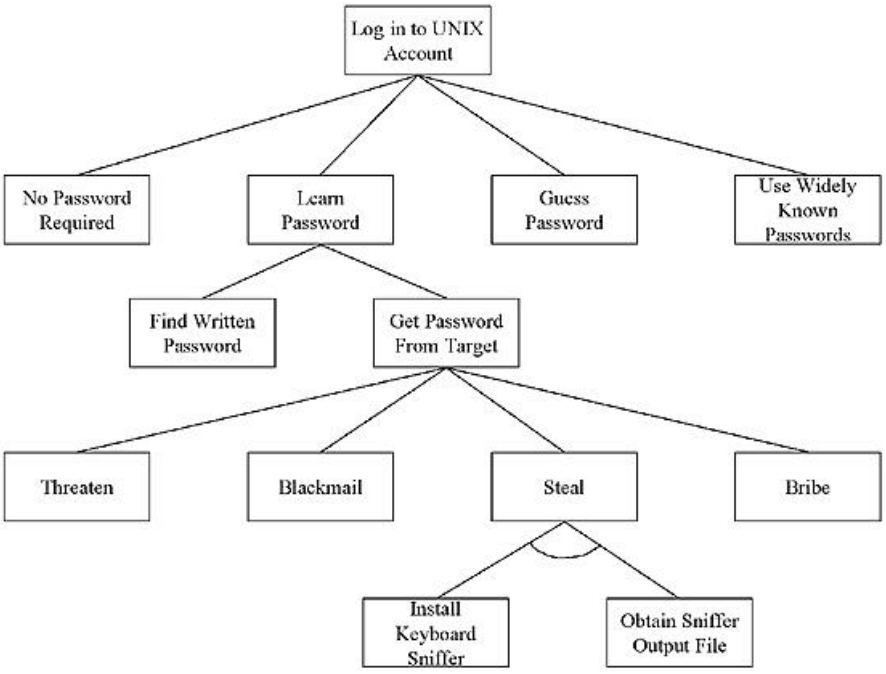


Figura 3.2 – Exemplo de Árvore de ataques[3]

[3] http://gemsres.com/photos/story/res/43842/fig4.jpg

3.1.2 Modelos baseados em rede

Os modelos baseados em rede têm como inspiração as redes Petri e surgiram após serem apontadas algumas falhas na estrutura hierárquica dos modelos baseados em árvore. Este modelo de ataques tem como principal proposta separar eventos de objetivos, sendo que os objetivos são representados numa estrutura estática e eventos ou ações correspondem as transições no modelo.

Embora à primeira vista este modelo de ataques não permita uma análise atómica dos seus componentes, permite ao analista aprofundar mais detalhadamente os componentes em cada um dos objetivos [8].

Existe também ainda alguma controvérsia referente aos modelos de ataques baseados em redes. Ainda não existem provas significativas de que estes possam ser de algum modo melhores que os modelos baseados em árvore, que são neste momento os modelos de ataques implementados em maior escala.

3.1.3 Modelo baseado na descrição de padrões de segurança

Esta forma de representação visa representar as ameaças e os ataques em forma de texto. Esta é uma forma muito utlizadas por especialistas em segurança de informação pois permite uma análise mais detalhadas de cada componente do ataques. O facto de estar descrita em texto simples tem a enorme vantagem de facilitar a interpretação, embora a falta de semântica e regulamentação na sua descrição possa também tornar complexa a sua interpretação.

Como principal inconveniente, este tipo de representação prende-se com o facto de não estar regulamentada, e o facto de ser apenas texto torna impossível a automação dos modelos de ataques [8].

3.2 Soluções existentes

Existe um vasto leque de soluções no que toca a modelação de ataques. Serão abordadas de seguida apenas as soluções que ao longo da dissertação se mostraram mais relevantes.

Paruchuri en Saini et al. 2008 apresentou um método de criação de árvores de ataques, assim como propôs uma ferramenta de criação de árvores de ataques, onde o objetivo principal se encontra na raiz da árvore e as suas folhas corresponde a sub-tarefas no ataque ou a ações tomadas para avançar na árvore, como pode ser visto na Figura 3.3. [11]

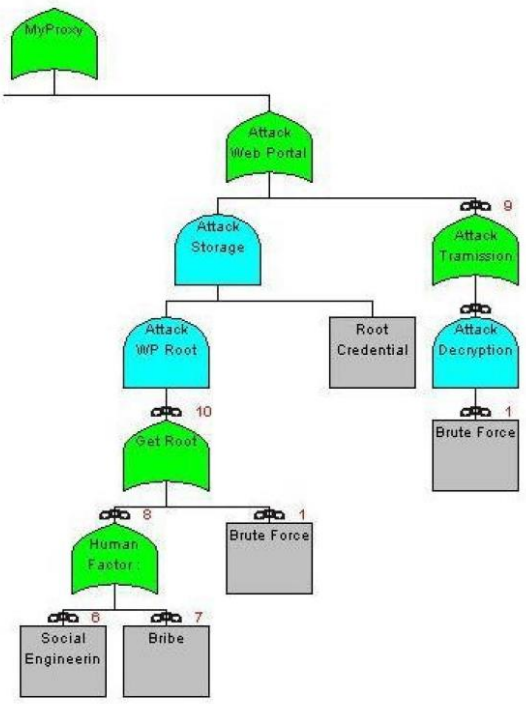


Figura 3.3 – Árvore de ataques a um portal web [11]

Por outro lado Dawkins en Larson et al. 2002 [2] propôs um modelo de ataque em árvore organizado por camadas tentando deste modo contornar as ambiguidades

existentes nas árvores de ataques. Este tipo de modelo, que difere da estrutura de uma arvore de ataques tradicional pode se visto na Figura 3.4.

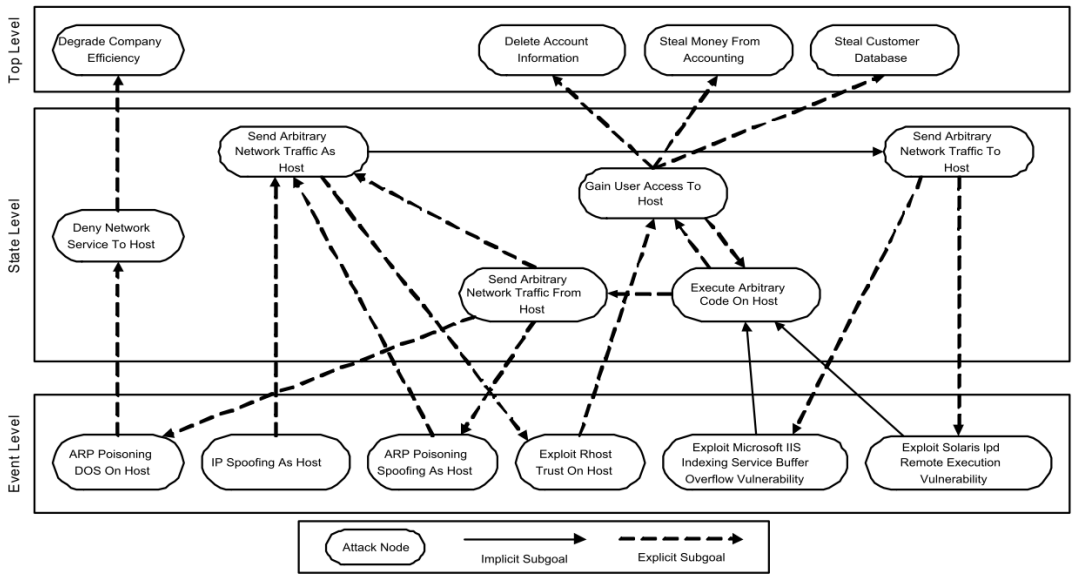


Figura 3.4 - Modelo estruturado em camadas [2]

Em contraponto, os autores Saber en Bouchentouf (2010) após análise de mais de 70 tipos de malware da lista *Common Weakness Enumeration* (CWE) [10] propuseram que independentemente da experiência do atacante, motivação ou do malware automático, a evolução dos ataques flui de acordo com o modelo apresentado na Figura 3.5.

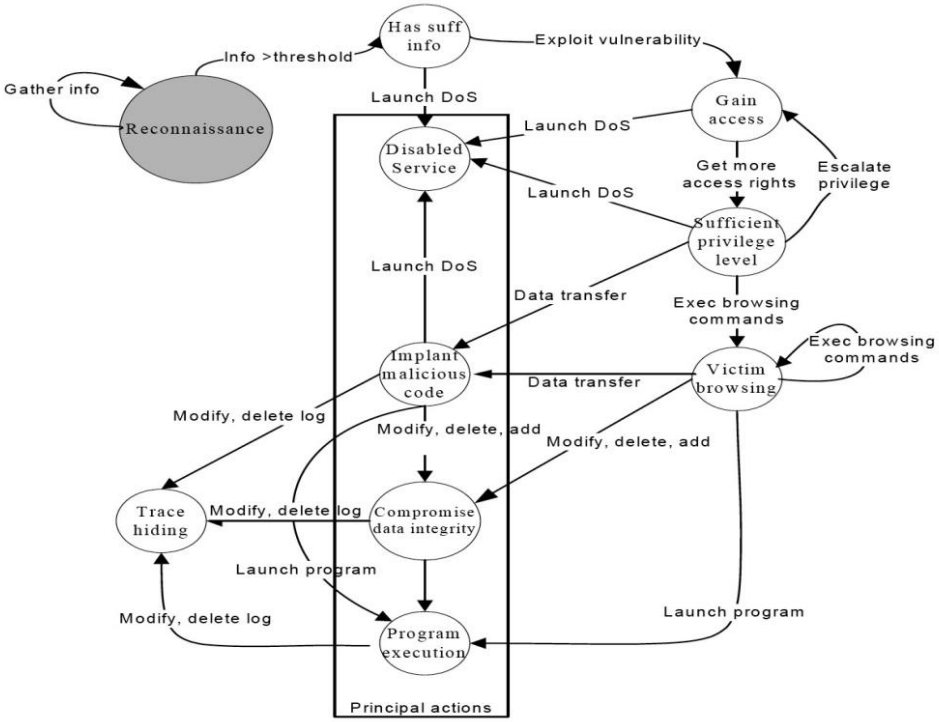


Figura 3.5 – Modelo de ataques [3]

Após cuidada análise, o modelo que se mostrou ser mais adequado para o caso em estudo foi o "*Execution Patterns*"[3], pois tem por base a análise comportamental do atacante e o pressuposto que o atacante segue um padrão lógico e previsíveis no decorrer da sua ação. Este padrão pode ser condicionado por fatores como o conhecimento e a motivação, mas mantendo-se sempre dentro do modelo previsto [3].

O modelo que pode ser observado na Figura 3.5 e descrito em seguida, representa etapas de ação do atacante ou do malware na rede e o que sucede caso o atacante tenha sucesso nessa ação.

O modelo apresenta as seguintes fases:

- **R: Reconnaissance** : Etapa essencial num ataque informático. É nesta etapa que o atacante recolhe a maior quantidade de informação possível, como o Sistema

Operativo utilizado e serviços disponíveis, possibilitando a determinação das suas vulnerabilidades e ferramentas necessárias para a realização do ataque.

- **GA: Gain Access** : De uma forma geral, o objetivo do ataque informático é ganhar acesso a um determinado sistema, para comprometer os seus recursos. Esta etapa consiste em utilizar técnicas/falhas do sistema que permitam acesso ao mesmo.

- **PE: Privilege escalation**: Embora muitas vezes o atacante tenha acesso à sua máquina alvo, não raras vezes este não tem os privilégios de administração para continuar os ataques. Por esse motivo é necessário recorrer a técnicas específicas que permitem aumentar os seus privilégios.

- **VB: Victim Browsing** : Após obter acesso a um determinado alvo, com os privilégios necessários, o atacante pode ter como principal alvo espiar a vítima. Nesse caso, o atacante pode recorrer a várias técnicas que permitam registar o comportamento da vítima, assim como credenciais de acesso.

- **HT: Hide Traces:** Todo os ataques a sistemas de informação deixam vestígios que ficam registados nos logs do Sistema Operativo ou do sistema de deteção de intrusões. Não raras vezes, o atacante no final do ataque tenta camuflar o seu rasto como forma de proteção. Esta etapa engloba as técnicas de camuflagem de vestígios, de modo a tornar a pesquisa forense mais difícil ou até mesmo impossibilitar a mesma.

- **Principal action**: Este ponto não é uma etapa por si só, mas sim um conjunto de etapas:

- **EP: Execute Program**: Grande parte do malware, após instalado permite receber comandos específicos do seu atacante. Esta etapa visa englobar as técnicas que permitem ativar/executar ordem nos *malwares* já instalados.

- **IMC: Implant Malicious Code :** Muitas vezes os ataques têm por objetivo a instalação de malware na vítima. Esta etapa engloba as técnicas de instalação de código malicioso.

- **CDI: Compromise Data Integrity:** Conjunto de técnicas que consistem em comprometer a integridade dos dados. Estas técnicas podem apenas recolher dados, ou atuar de forma mais invasiva, alterando/destruindo dados.

- **DoS: Denial of Service:** Um dos principais flagelos da Internet atual, consiste em tornar os serviços permanentemente ou temporariamente indisponíveis. Este grupo visa abranger os ataques que tem por objetivo comprometer a disponibilidade do sistema.

3.3 Bibliotecas de ataques

Uma biblioteca de ataques é uma ferramenta útil, descrevendo ataques alvo de estudo. É de referir que as bibliotecas podem diferir ao nível do detalhe de descrição do ataque.

Este tipo de ferramenta tem por objetivo o apoio na categorização dos ataques e a análise comportamental dos mesmos. Não descreve detalhadamente o funcionamento do ataque mas oferece uma estrutura que ajuda na sua compreensão e organização. [14].

Nos subtópicos seguintes serão descritas algumas das principais bibliotecas de ataques utilizadas atualmente.

3.3.1 Common Attack Pattern Enumeration and Classification (CAPEC)

É uma lista de acesso livre, com um total de 463 ataques diferentes, aquando a versão 2.6. Cada ataque contém informação detalhada sobre o mesmo, embora seja importante realçar que nem todos contêm o mesmo nível de detalhe.

Um ataque que contenha o seu nível de detalhe ao máximo, o qual pode ser visto um exemplo no Apêndice C, deve conter os seguintes itens:

- Typical severity
- A description, including:
  - Summary
  - Attack execution flow
- Prerequisites
- Method(s) of attack
- Examples
- Attacker skills or knowledge required
- Resources required
- Probing techniques
- Indicators/warnings of attack
- Solutions and mitigations
- Attack motivation/consequences
- Vector
- Payload
- Relevant security requirements, principles and guidance
- Technical context

É de realçar que o CAPEC tem duas estruturas hierárquicas bem definidas, que permitem uma rápida e fácil busca pelo tipo e ataque desejado.

A primeira estrutura hierárquica, representada na Figura 3.6, está organizada de acordo com os mecanismos de ataque utilizados para comprometer os sistemas.

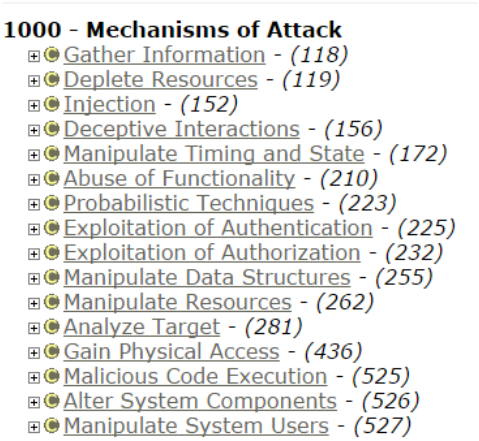
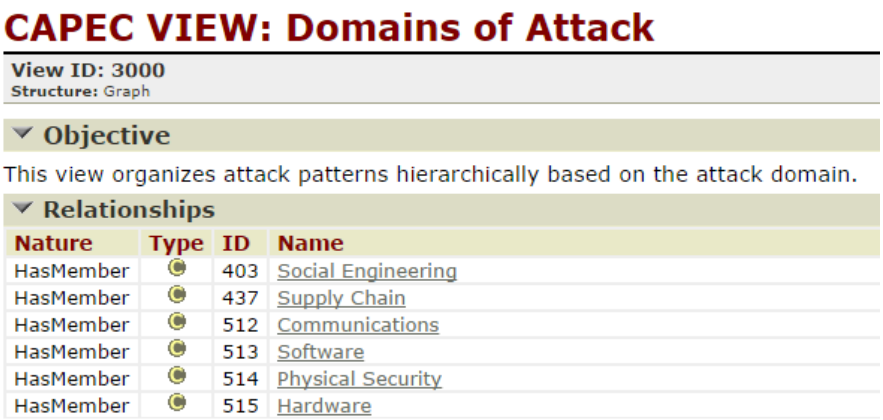


Figura 3.6 - Estrutura Hierárquica CAPEC[4]

A segunda estrutura hierárquica de representação do CAPEC é baseado no domínio dos ataques, como pode observado na Figura 3.7.



CAPEC VIEW: Domains of Attack

View ID: 3000
Structure: Graph

Objective

This view organizes attack patterns hierarchically based on the attack domain.

Relationships

Nature	Type	ID	Name
HasMember		403	Social Engineering
HasMember		437	Supply Chain
HasMember		512	Communications
HasMember		513	Software
HasMember		514	Physical Security
HasMember		515	Hardware

Figura 3.7 - Estrutura Hierárquica CAPEC[5]

[4] http://capec.mitre.org/
[5] http://capec.mitre.org/

Visto tratar-se de uma lista bastante extensa, contendo vários tipos de ataques que não são relevantes para este projeto, optou-se por criar uma lista hierárquica dos ataques mais relevantes para ser estudada no modelo de ataques. Essa lista encontra-se disponível para consulta no Apêndice A.

3.3.2 STRIDE

STRIDE é um acrónimo para *Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service e Elevation of Privilege* [14].

É um mecanismo de catalogação de ameaças. Alguns especialistas defendem que este não é uma das abordagens mais adequadas, devido ao facto que muitas vezes, em ataques mais complexos, o mesmo ataque pode ser englobado em mais que uma categoria deixando o investigador com resultados ambíguos.

No entanto, embora apresente alguns problemas, esta forma de catalogação é uma boa forma de estudo das vulnerabilidades e uma ferramenta útil de apoio para determinar formas de mitigar essas vulnerabilidades. O seu ponto forte centra-se no facto de que a cada ponto que representa uma ameaça, corresponde uma propriedade do sistema que é violada, como pode ser observado na Tabela 3.1.

Ameaça	**Propriedade violada**
Spoofing	Autenticação
Tampering	Integridade
Repudiation	Não repudio
Information Disclosure	Confidencialidade
Denial of Service	Disponibilidade
Elevation of Privilege	Autorização

Tabela 3.1 - STRIDE Threats [14]

3.3.3 OWASP TOP 10

Esta biblioteca visa educar e alertar os especialistas na área de segurança para as vulnerabilidades e principais ataques que ocorreram no passado ano de 2013. Para elaborar esta lista foram recolhidos dados fornecidos por sete empresas especializadas na área de segurança de informação.

Na versão de 2013 as dez principais vulnerabilidades de ataque foram as seguintes: [6]

1. Injection
2. Broken Authentication and Session Management
3. Cross-Site Scripting
4. Insecure Direct Object References
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Missing Function Level Access Control
8. Cross Site Request Forgery
9. Components with Known Vulnerabilities
10. Invalidated Requests and Forwards

3.3.4 Pcap attack library

Biblioteca de ataques publicada por Burroughs en Engebretson et al. 2010 [19]. Esta biblioteca visa oferecer uma lista de ataques baseada no CAPEC de uma forma organizada.

Segundo os autores Burroughs en Engebretson et al. 2010 as bibliotecas para teste de IDS existentes tem problemas de desorganização, contento muito tráfego da rede que não é relevante para os ataques. Para colmatar esta falha, os autores recolheram o

[6] https://www.owasp.org/index.php/Top_10_2013-Top_10

tráfego apenas entre a máquina atacante e a vítima, podendo assim reproduzir os ataques, colocando novamente o tráfego na rede.

Adicinalmente e por forma a compreender melhor o comportamento do ataque, os investigadores recolhem os eventos gerados pelo SNORT IDS que podem ser utilizados para detetar pacotes específicos no ataque [19].

O processo de captura de tráfego pode ser observado na Figura 3.8.

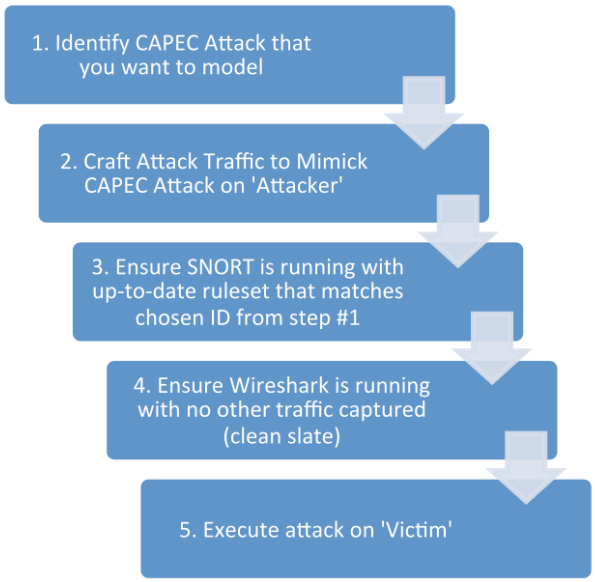


Figura 3.8 – Processo de captura de tráfego [19]

3.4 Ferramentas

Neste tópico, serão descritas todas a ferramentas necessárias para o desenvolvimento de um ambiente controlado, assim como os sistemas de deteção de intrusões utilizados para recolher eventos relevantes para as fases de decisão do modelo de ataques.

3.4.1 GNS3[7]

O GNS3 é um *software opensource* e multiplataforma que permite a simulação de redes complexas, sem a necessidade de componentes físicos. Graças à sua integração com o Virtualbox torna-se possível a criação de redes em ambiente controlado para testes de segurança.

Esta ferramenta de simulação de rede foi escolhida por ser gratuita e ter um ambiente gráfico muito intuitivo, permitindo ligar vários componentes.

Na Figura 3.9 pode ser vista uma rede com três máquinas e vários componentes de rede como *routers* e *firewall*.

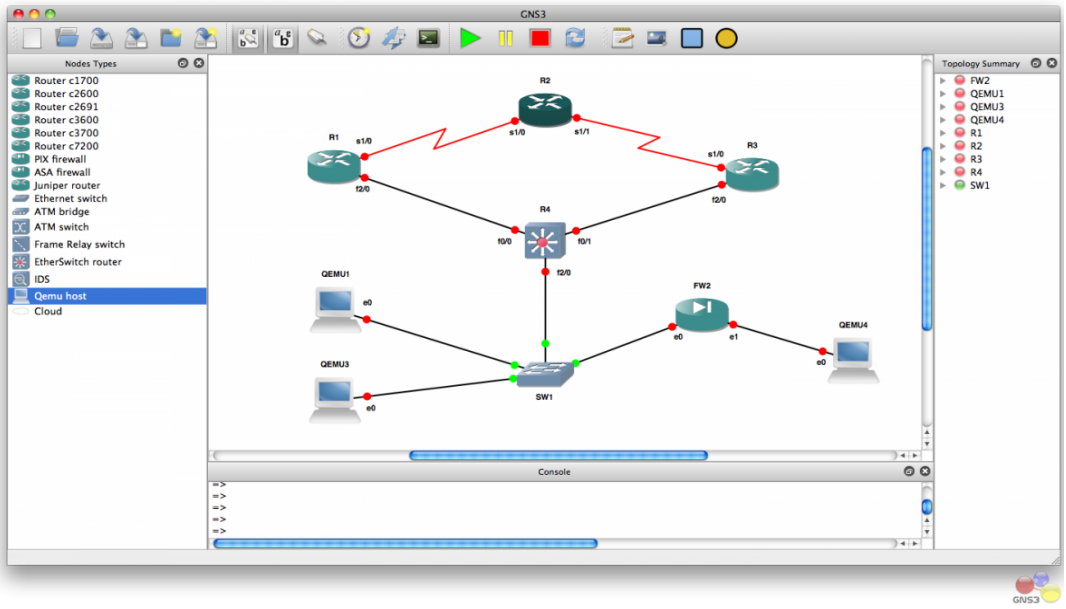


Figura 3.9 – Exemplo de topologia criada na Ferramenta GNS3[4]

3.4.2 Oracle Virtualbox[8]

Ferramenta poderosa de virtualização de máquinas que permite emular uma máquina virtual com um sistema operativo, para ligar à topologia.

[7] http://www.gns3.net/
[8] https://www.virtualbox.org/

Pelas dimensões da rede de simulação, tornou-se inviável ter máquinas reais ligadas a uma rede real, devido aos custos elevados que acarretaria. A escolha recaiu sobre esta ferramenta de virtualização, que quando confrontada com concorrentes diretos obteve vantagem, por ser gratuita e de fácil integração com o sistema de simulação de redes utilizado.

3.4.3 Wireshark[9]

Ferramenta de análise e captura de tráfego mais utilizada em todo o mundo em grande parte devido à sua simplicidade de utilização ao facto de ser gratuita e de se tratar de uma ferramenta multiplataforma.

O *software* necessita de ser instalado numa máquina. Após isso, oferece um vasto leque de ferramentas que permitem recolher e analisar o acontece na rede. Recorrendo aos seus filtros pode-se facilmente, determinar os protocolos que estão a ser utilizados ou as aplicações que geram maior quantidade de tráfego [20].

3.4.4 Sistemas Operativos

No presente subtópico serão apresentados os diferentes sistemas operativos instalados, nas diferentes máquinas virtuais, que visam colmatar as necessidades específicas no ambiente de simulação.

3.4.4.1 Kali Linux[10]

Distribuição Linux criada com o apoio da comunidade ligada às cyber segurança, com o propósito de oferecer um vasto leque de ferramentas de penetração, juntas num único

[9] https://www.wireshark.org/
[10] http://www.kali.org/

sistema operativo. Dentro do mesmo sistema operativo os profissionais tem todas as ferramentas que necessitam, incluindo as que garantem a confidencialidade da máquina, já que este sistema operativo fornece ferramentas de criptografia, que cifram todo o sistema operativo.

Este sistema operativo foi escolhido porque como referido anteriormente, contem as ferramentas fundamentais para fazer os ataques necessários para teste do modelo.

3.4.4.2 Windows XP SP2

O sistema operativo Windows XP SP2 é um sistema operativo da Microsoft com uma quota significativa de mercado, que deixou de receber atualizações de segurança no presente ano de 2014 e por esse motivo tornou-se um alvo ideal para testes.

O principal fator de opção por esta versão específica do sistema operativo centrou-se no livro "*Professional Penetration Testing*" [21] que apresenta uma lista de ataques muito completa, com um bom tutorial para explorar as vulnerabilidades deste sistema operativo.

3.4.4.3 Ubuntu Server 14.04.1 LTS

Sistema operativo Linux instalado com o propósito de alojar serviços web vulneráveis para testes. Este sistema operativo não tem propósito de ser atacado, mas sim de oferecer recursos passiveis de serem alvos de ataque.

3.4.5 Aplicações/Serviços

Nos dias atuais, é cada vez mais importante estar atualizado sobre das vulnerabilidades existentes e disponíveis no mercado. Como nem sempre é possível

fazer os testes/treino em ambiente real devido à seriedade dos dados envolvidos ou devido à falta de permissão para os testes, é necessário fornecer aos recém chegados à área de cyber segurança um conjunto de ferramentas que permitam fazer os seus testes em ambientes controlados sem provocar danos em terceiros.

De seguida estão descritas algumas dessas ferramentas.

3.4.5.1 WebGoat[11]

Ferramenta desenvolvida pelo OWASP, em java, sendo uma multiplataforma com o propósito de fornecer a quem esta a iniciar-se na área de testes de penetração de rede, um vasto leque de serviços com vulnerabilidades que podem ser testadas. O projeto está desenhado para oferecer pequenos desafios que vão aumentando de dificuldade. Entre outras vulnerabilidades encontramos as seguintes:

- Cross-site Scripting (XSS)
- Access Control
- Thread Safety
- Hidden Form Field Manipulation
- Parameter Manipulation
- Weak Session Cookies
- Blind SQL Injection
- Numeric SQL Injection
- String SQL Injection
- Web Services
- Fail Open Authentication
- Dangers of HTML Comments

[11] https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project

Esta ferramenta foi incorporada no projeto, pois como oferece uma lista de vulnerabilidades a serem testadas, facilitando a exploração dos eventos para integração no modelo.

3.4.5.2 Damn vulnerable web app (DVWA)[12]

Trata-se de uma aplicação web no sentido tradicional, ou seja, com uma base de dados em MySQL, PHP e a ser executada sobre a plataforma XAMPP.

Esta aplicação tem uma vasta lista de vulnerabilidades que podem ser exploradas. Por forma a conseguir recolher eventos relevantes para o modelo de ataques em estudo.

O DVWA inclui grande parte das vulnerabilidades que pertencem ao TOP 10 da OWASP. Algumas das vulnerabilidade que podem ser encontradas no DVWA [22] são:

- **Brute Force**: Permite testar com ferramentas próprias se os utilizadores contem palavras-chave inseguras.

- **Command Execution**: Executa comandos remotamente no Sistema Operativo sem permissões.

- **Cross Site Request Forgery (CSRF)**: Permite que o atacante altere as credenciais de administração.

- **SQL Injection: Injeção de querys SQL**: Compromete a confidencialidade da base de dados.

- **Insecure File Upload**: Permite injetar ficheiros remotamente no servidor.

- **Cross Site Scripting (XSS)**: Permite injetar scripts na aplicação web.

[12] http://www.dvwa.co.uk/

3.4.6 Metasploit

O *Metasploit* não é uma simples ferramenta, é uma *framework* que oferece aos especialistas em segurança de informação um conjunto de componentes que permite realizar ataques complexos, de forma simples e eficaz [23].

A escolha desta *framework* foi feita devido ao facto de ser uma das *framework* mais utilizadas na área de testes de penetração. No livro "Metasploit The Penetration Tester's Guide" [23] o autor Kennedy en Gorman et al.2011 apresenta um guia muito completo, que se revelou bastante útil no desenvolvimento dos ataques, para teste no ambiente controlado.

3.4.7 Sistema de deteção de intrusões utilizados

Ao contrário do tópico 2.2, que apenas visava explicar os tipos de sistemas de deteção de intrusões existentes, assim como o seu propósito/modo de funcionamento neste ponto são descritos com mais detalhe os sistemas de deteção de intrusões utilizados no decorrer da simulação.

3.4.7.1 SNORT[13]

Embora não seja o único NIDS, é sem dúvida o mais utilizado, não apenas por ser gratuito, pois existem outros sistemas de deteção de intrusões que também o são, mas devido à sua grande comunidade que contribui para o desenvolvimento da vasta biblioteca de regras de deteção de ataques [18]. O SNORT contem um conjunto de componentes que executam tarefas individuais e que formam o NIDS quando ligadas como um todo, como pode ser observado na Figura 3.10.

[13] https://www.snort.org/

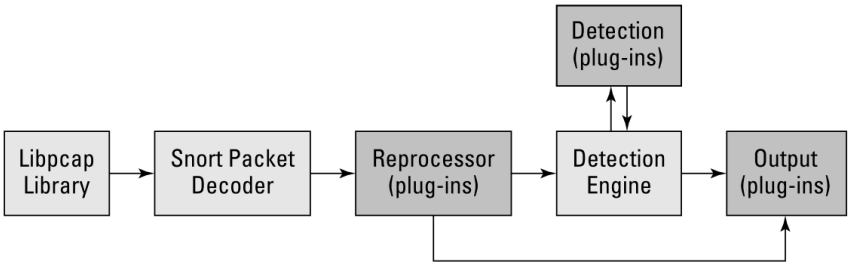


Figura 3.10 – Fluxo de processamento pacotes SNORT [18]

Sempre que pacotes de rede, que possam ser considerados pertencentes a um ataque, são detetados na rede é gerado um alerta, como pode ser visto um exemplo na Figura 3.11

```
02/26-17:59:01.635549  [**] [1:2003:2] MS-SQL Worm
          propagation attempt [**] [Classification: Misc
          Attack] [Priority: 2] {UDP} Y.Y.250.124:1162 ->
          X.X.2.27:1434
```

Figura 3.11 – Exemplo alerta SNORT [18]

3.4.7.2 OSSEC[14]

O OSSEC é um HIDS que oferece um sistema de monitorização centralizado permitindo a recolha de logs e eventos em apenas uma máquina. É composto por vários componentes espalhados pela rede, sendo que esses componentes podem ser distinguidos em *Manager* e Agente.

O Manager é o servidor que recolhe todos logs enviados pelos agentes e age de acordo com o que o gestor pré-estipulou, podendo enviar alertas via correio eletrónico, ter uma postura ativa na prevenção ou apenas registar os logs para posterior consulta.

[14] http://www.ossec.net/

Ao contrário do Manager que é único na rede, existe um agente por cada máquina da rede, sendo que o *software* do agente é responsável apenas por recolher os logs das máquinas e pelo envio seguro para o servidor, deixando a interpretação das regras e alertas para o mesmo. Como o agente executa uma tarefa tão simples, o seu processamento é imperceptível a nível de consumos de recursos na máquina onde está a ser executado.

Pode ser observado abaixo um exemplo de uma rede com vários agentes, na Figura 3.12

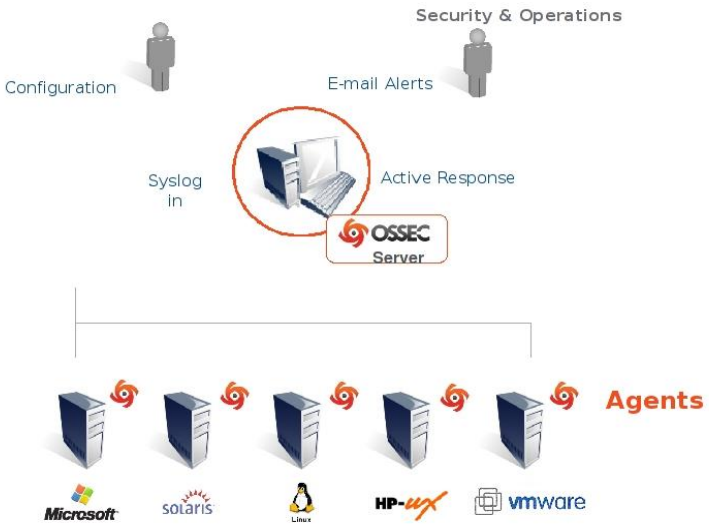


Figura 3.12 – Exemplo de uma arquitetura OSSEC[15]

[15] http://www.ossec.net/wp-content/uploads/2012/04/ossec-arch2.jpg

Capítulo 4

4 Implementação e teste do modelo de ataques

Neste capítulo serão abordados os requisitos necessários para o desenvolvimento de um modelo de ataques eficaz, seguido da escolha da catalogação e dos ataques a serem testados.

Será explicada a topologia criada para testes em ambiente controlado, bem como a metodologia de recolha de eventos e *datasets* para cada um dos ataques.

Ainda neste capítulo, serão apresentados os ataques testados juntamente com os eventos recolhidos em cada um dos mesmos.

Por fim, será também abordado a necessidade de criação de uma ferramenta capaz de reintroduzir o tráfego recolhido novamente na rede.

4.1 Requisitos do modelo de ataques

Grande parte dos modelos propostos, como o modelo de Mahoney en Chan et al. 2002 [9], são modelos de ataques que têm por base apenas eventos gerados pela rede. Outros modelos visam apenas modelar ataques a *software* ou máquinas específicas.

Visto na segurança de informação não ser sensato desprezar nenhum dos componentes referidos, a recolha deve ser feita tanto ao nível do host como ao nível da rede, recorrendo as ferramentas previamente descritas no tópico 3.4 da presente dissertação.

Este é um dos pontos mais importantes na recolha de eventos para o modelo de ataques projectado.

Alguns modelos como os propostos por Cheung en Fong et al. 2003 [1] e Dahl en Wolthusen et al. 2006 [24] são modelos que dependem da topologia de rede. Como a dependência da topologia é, de um modo geral, um grande entrave na criação de modelos de ataques foi determinado como principal objetivo da presente dissertação o foco no desenvolvimento de um modelo genérico para testes de sistemas de deteção de intrusões. Um dos objetivos é a abstração de topologias de rede, podendo o modelo ser implementado em todos os géneros de ataques.

Tal como referido por Gadelrab en Kalam et al. 2008 [3], os ataques, sejam eles automáticos ou não, tem tendência para alterar suas variáveis comportamentais por modo a contornar as assinaturas dos sistemas de deteção de intrusões. Por esse motivo e por modo a colmatar os problemas de excesso de assinaturas utilizadas pelos sistemas de deteção de intrusões, referidos no tópico 1.1, torna-se essencial que o modelo tenha uma estrutura o mais genérica possível, permitindo criar uma assinatura genérica para um determinado grupo de ataques.

De acordo com o estudo apresentado por Burroughs en Engebretson et al. 2010 [19], uma das grandes lacunas nos testes de sistemas de deteção de intrusões é a falta de tráfego organizado, que possa ser reproduzido novamente, para repetição dos eventos e consequente análise e verificação se os sistemas de deteção de intrusões são capazes de gerar esses eventos novamente. Por esse mesmo motivo, um outro objetivo prioritário é a criação de um *dataset* gerado durante um determinado ataque. Esse tráfego deve ser apresentado da forma mais simplificada possível, ou seja, deve conter única e exclusivamente o tráfego direcionado do atacante para a vitima e vice-versa. Esse *dataset* já foi sugerido no estudo apresentado por Burroughs en Engebretson et al. 2010, mas por motivos indeterminados não foi continuado sendo que atualmente é ainda utilizado na avaliação de sistemas de deteção de intrusões *datasets* recolhidos nos anos de 1998 e 1999 numa ação conjunta entre o MIT e o DARPA[16] [33].

Por fim, seria uma mais-valia para o projeto, o desenvolvimento de um software que permita a reintrodução do tráfego organizado recolhido na rede, para possibilitar a

[16] http://www.darpa.mil/default.aspx

realização de mais testes reais aos sistemas de deteção de intrusões. Um software semelhante ao exposto, já foi proposto pelos autores Burroughs en Engebretson et al. 2010 [19] no Engebretson en Cronin et al. 2010 [25] não tendo sido possível determinar junto da instituição que os autores representam se esse mesmo software foi continuado.

4.2 Modelo proposto

O modelo proposto, que pode ser observado na Figura 4.1, embora tenha sido baseado no "Execution Patterns"[3] necessitou de pequenas modificações. Isto, de modo a aperfeiçoa-lo, para ir de encontro aos objetivos propostos e de forma a tornar o modelo mais compreensível e inteligível.

Na proposta final para este projeto, tornou-se mais simples a compreensão do modelo, devido à reestruturação e diferenciação entre fases e ações, as quais se apresentavam de forma um pouco ambíguas no modelo original.

Outra alteração relevante ao modelo original prende-se com os denial of service (*DoS)*. No modelo original os autores deram elevada relevância ao *DoS*, podendo basicamente ser executado em qualquer fase do modelo de ataque. Embora os ataques por *DoS* tenham tido um crescimento significativo, tanto em número como em impacto nos últimos anos, ao longo do desenvolvimento da dissertação tornou-se claro que estes deveriam representar o mesmo nível de importância que qualquer outro tipo de ataque.

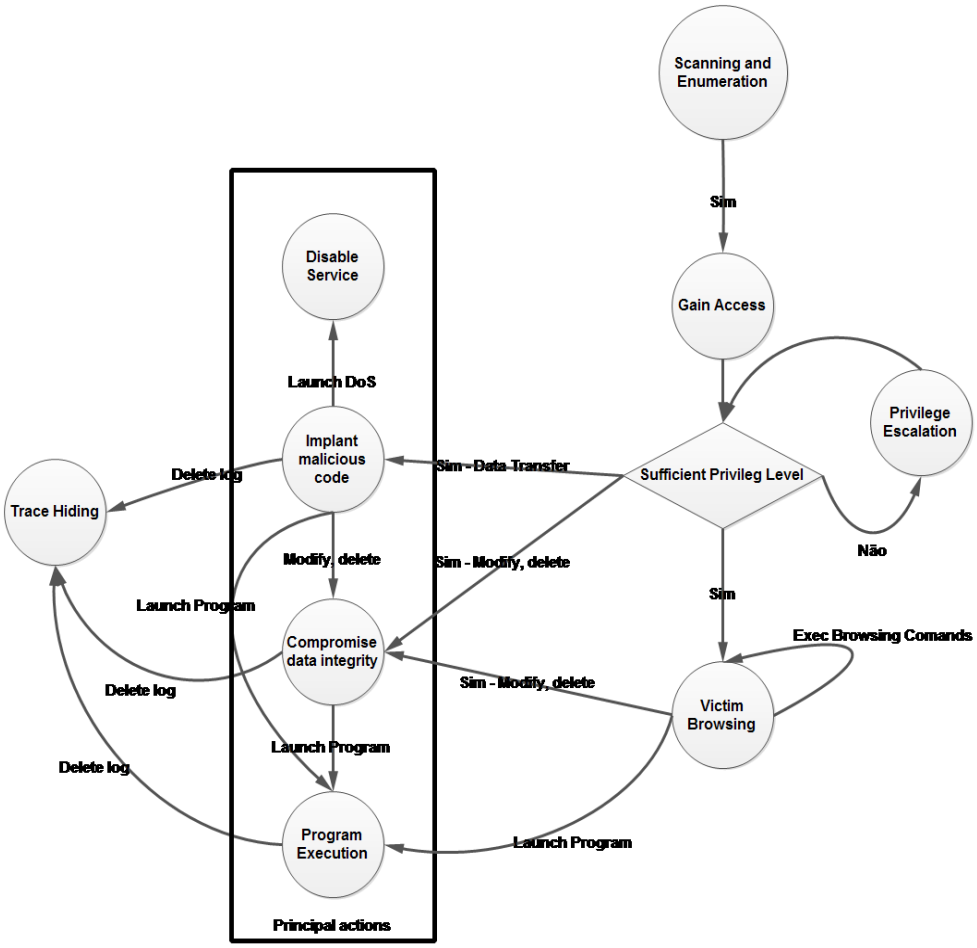


Figura 4.1 – Modelo Proposto

O modelo proposto, representado na Figura 4.1, encontra-se dividido em 9 fases, como explicado no ponto anterior. Cada ataque estudado irá enquadrar-se numa fase de decisão, sendo que um ataque mais complexo pode passar por várias fases de decisão no decorrer do tempo. Para facilitar a recolha de eventos e a perceção dos ataques, os ataques complexos devem ser divididos em ataques mais simples, formando um ataque complexo como um todo.

Por exemplo, um atacante que pretenda furtar informações de uma base de dados, deverá fazer um *scanning* da rede em primeiro lugar, de seguida conseguir os privilégios necessários para o ataque e posteriormente atacar a base de dados. Deste modo o atacante necessita de, pelo menos, passar por três fases de decisão do modelo, sendo que cada uma despoleta eventos próprios.

4.3 Catalogação e escolha dos ataques

A catalogação de ataques em grupos, tal como referido anteriormente, pode ser uma tarefa difícil, mas quando realizada ajudará a avançar significativamente na investigação.

Neste caso, a técnica de catalogação mais adequada para a escolha dos ataques a serem testados no ambiente controlado, foi a técnica de catalogação apresentada pelo especialista Shostack A. et al. 2014 em *Threat Modeling* da Microsoft no seu livro "*Threat Modeling Designing for Security*" [14]. Esta técnica prevaleceu em relação as outras, pelo facto de se adequar bem ao modelo, necessitando apenas de pequenos ajustes para abranger todas as fases de decisão do modelo de ataques proposto. Por exemplo, um ataque a uma base de dados que visa alterar ou destruir dados da mesma, deverá estar catalogado no grupo "*Tampering*".

Por forma a criar um modelo de ataque, o mais completo possível, e visto não ser possível por uma razão de tempo e recursos testar todos os ataques apresentados no Apêndice A optou-se por utilizar alguns ataques tentando assim abranger todas as fases de decisão do modelo.

Como primeiro passo para fazer a escolha dos ataques a testar pretendeu-se catalogar os ataques de forma a conseguir agrupa-los. Uma das técnicas de catalogação (STRIDE) mais utilizadas para agrupar ataques/vulnerabilidades baseia-se na ameaça que o ataque representa, como pode ser visto no exemplo abaixo.

- Spoofing:
    - Buffer overflow (CAPEC-100)
    - Password Brute Forcing (CAPEC-49)
- Tampering:
    - Forceful Browsing (CAPEC-87)

    - SQL Injection (CAPEC-66)
- Repudiation:
- Information Disclosure:
    - Men in the middle (CAPEC-94)
- Denial of Service:
    - HTTP DoS (CAPEC-469)
    - TCP Flood (CAPEC-482)
    - Pointer Attack (CAPEC-129)
- Elevation of Privilege:
    - Restful Privilege Elevation (CAPEC-58)

Embora não pertença ao grupo inicial do STRIDE, foi também adicionada a componente de reconhecimento, como pode se visto no exemplo abaixo. Pois o reconhecimento da rede alvo é uma das componentes essenciais de um ataque.

- Reconnaissance :
    - Footprinting/Enumeration (Capec-169)
    - Active OS Fingerprinting (CAPEC-312)

4.4 Ambiente controlado para testes

O ambiente controlado para testar os ataques escolhidos no tópico 4.3 é o ponto central de todo o projeto. Nesse ambiente será possível validar e recolher eventos e tráfego relevantes para o desenvolvimento do projeto. Visto que a criação de um ambiente controlado requerer um vasto e variado leque de máquinas é inviável a criação desse sistema em máquinas reais.

Na Figura 4.2 está representada a topologia desenhada neste projeto, para os testes em ambiente controlado.

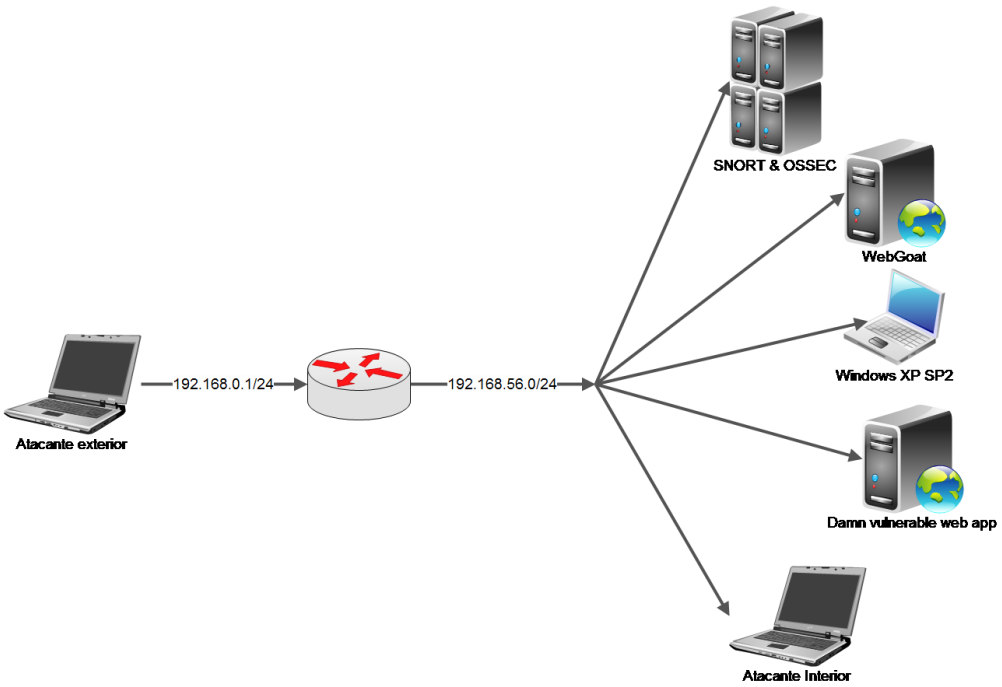


Figura 4.2 – Topologia utilizada no ambiente controlado

Para a tornar os ataques mais realistas, foram utilizadas duas redes diferentes, permitindo a existência de um atacante externo à rede e um atacante interno à rede.

A rede alvo foi projetada com um total de 6 máquinas, sendo que uma delas representa um atacante interno. Desta forma, é possível a realização de ataques em que é necessário acesso interno, como por exemplo o ataque *men in the middle*.

São utilizados dois servidores Web, cujos serviços específicos já foram descritos no subtópico 3.4.5 do presente documento.

Seguindo a recomendação do livro Metasploit The Penetration Tester's Guide et al. 2011 [23], foi também utilizada uma máquina com Windows XP SP2, de modo a realizar testes em máquinas sem serviço web simulado, abrangendo desta forma as máquinas de utilização diária, que representam uma grande percentagem da web atual.

Por último, mas não menos importante, foram utilizadas duas máquinas com os sistemas de deteção de intrusões referidos no subtópico 3.4.7, que visam recolher os eventos para aplicação no modelo de ataques. Na máquina que contem o SNORT também foi instalado o Wireshark, para recolha dos ficheiros PCAP que farão parte do *dataset* proposto.

4.5 Estratégia de recolha de eventos e de dataset

A recolha de eventos será realizada para cada um dos ataques apresentados no tópico 4.3. Esta recolha foi baseada na recolha apresentada por Burroughs en Engebretson et al. 2010 [19], sendo acrescentada a recolha de eventos de sistemas de deteção de intrusões baseados no *host*.

Como pode ser visto na Figura 4.3, o processo de geração e recolha de eventos está dividido em cinco etapas distintas.

A primeira etapa corresponde à escolha de um ataque da biblioteca de ataques CAPEC, que vá ao encontro da necessidade do estudo.

Na segunda e terceira etapa assegura-se que os dois sistemas de deteção de intrusões estão em funcionamento e prontos a gerar eventos para recolha.

De seguida, na quarta etapa garante-se que a recolha de tráfego pelo Wireshark está a funcionar e que os devidos filtros são aplicados, para garantir que o único tráfego apresentado é o tráfego entre a vítima e o atacante. Garante-se assim um tráfego limpo e passível de posterior reprodução.

Quando as três etapas anteriores foram executadas com êxito é então executado o ataque propriamente dito, e essa é a quinta e última etapa da geração de eventos.

Figura 4.3 – Processo de recolha de eventos e dataset

4.6 Lista de ataques testados

Neste tópico serão apresentados onze ataques da lista CAPEC apresentando os dados recolhidos em cada ataque.

Cada ataque testado encontra-se dividido em três fases essenciais:

- **Primeira fase**: breve descrição dos objetivos do ataque. Esta fase terá como base a descrição apresentada em cada um dos ataques, na lista do CAPEC.

- **Segunda fase**: técnica utilizada para fazer a simulação no ambiente controlado, descrevendo as ferramentas e as vulnerabilidades exploradas.
- **Terceira fase**: Descrição dos indicadores recolhidos durante o ataque. Essa última fase encontra-se dividida em três subfases, que em seguida se explanam:
    - Descrição dos eventos recolhidos pelo HIDS OSSEC, onde serão apresentados, se aplicável, os registos nos logs do sistema que ocorreram durante a simulação.
    - Lista de eventos despoletados durante o ataque no NIDS SNORT, ou seja, os eventos de rede que possam de algum modo ser mensuráveis.
    - Recolha do tráfego, recorrendo ao *software* de análise de tráfego de rede Wireshark e criação de uma tabela com esse tráfego, de modo a facilitar a identificação e padrões anómalos no tráfego de rede. Para facilitar a análise foi criada uma tabela comparativa com os dados dos 11 ataques assim como de tráfego não malicioso recolhido numa rede doméstica, essa tabela pode ser consultada no Apêndice E. Esta subfase visa também a recolha de ficheiros PCAP para criação de um *dataset* e também complementar os eventos recolhidos pelos IDS, que muitas vezes devido à falta de regras de deteção não são suficientes.

4.6.1 *Footprinting/Enumeration* (Capec-169)

Técnica inicial de recolha de informação utlizada pelos atacantes quando estudam uma rede alvo. O atacante tenta recolher informação sobre uma topologia de rede, de modo a identificar as máquinas ativas. Após este ataque é feito o *Fingerprinting*, cujo objetivo e funcionamento será explicado adiante.

4.6.1.1 Técnica

Neste caso recorreu-se à ferramenta nmap.

Nmap –sn –r –n <IP/MASK>

Onde a opção:

- **–sn**, visa desabilitar a verificação das portas Pois, neste caso, pretende-se apenas determinar os hosts que se encontram ligados;
- **-r**, força a verificação do nmap de forma ordenada, tornando o output mais percetível;
- **–n**, desabilita a resolução do DNS.

Os testes realizados no *fingerprinting* e no *footprinting* podem ser realizados em conjunto, retornando de imediato os hosts que se encontram online à medida que vai fazendo o reconhecimento da rede. Optou-se por realizar os testes em separado para melhor e mais fácil análise do tráfego gerado em cada ponto.

Como resultado do *scan* à rede encontram-se vários hosts ligados que podem ser testados na etapa seguinte.

4.6.1.2 Indicadores

Eventos OSSEC

Visto tratar-se se um evento de rede, não se registaram logs de sistema.

Eventos SNORT

- ICMP PING NMAP [1:469:1] [Classification: Attempted Information Leak] [Priority: 2]

Rede

Total Bytes	60322.98 bytes
Total pacotes	860 pacotes
Tempo total	34.472 segundos
Pacotes/segundo	24.947 pac/seg
Tamanho médio dos pacotes	70.143 bytes
Bytes/segundo	1749.891 bytes/seg
Protocolo	89.19% ARP

Tabela 4.1 – Estatísticas de tráfego do Capec-169

4.6.2 Active OS Fingerprinting (CAPEC-312)

Nesta etapa, após se determinar a máquina que se quer testar tentar-se-á determinar o Sistema Operativo, serviços instalados e portas abertas.

Se por alegoria, o *footprinting* fosse procurar uma casa para assaltar, o *fingerprinting* poderia ser considerado quem procura de portas ou janelas abertas para entrar nessa casa. [26]

4.6.2.1 Técnica

A ferramenta utilizada foi novamente o nmap.

Nmap –O – sV – sU – sS –n <IP>

Onde a opção:

- **–O** utiliza o *scanning* para retornar o sistema operativo instalado na máquina,
- **-sV** verifica os serviços e respetivas versões instaladas nas portas,
- **-sU** realiza também a verificação de portas UDP
- **–sS** visa não finalizar o ping, terminando assim a ligação e reduzindo a hipótese de deteção.

4.6.2.2 Indicadores

Eventos OSSEC

Visto tratar-se um evento de rede, não se registaram logs de sistema.

Eventos SNORT

- ICMP PING NMAP [1:469:1] [Classification: Attempted Information Leak] [Priority: 2]

Rede

Total Bytes	316177 bytes
Total pacotes	4533 pacotes
Tempo total	110.762 segundos
Pacotes/segundo	28.625 pac/seg
Tamanho médio dos pacotes	70 bytes
Bytes/segundo	2854.553 bytes/seg
Protocolo	52.31% TCP 22.30% TCMP 25.13% UDP

Tabela 4.2 - Estatísticas de tráfego do Capec-312

4.6.3 Buffer overflow (CAPEC-100)

Esta técnica visa explorar falhas na alocação dos *buffers* (normalmente em software), fazendo o *software* falhar. Abrem-se assim portas, deixando a máquina vulnerável, o que permite o acesso remoto e controlo da conta que esta a utilizar o mesmo.

4.6.3.1 Técnica

Recorrendo a uma falha existente num reprodutor de músicas para Windows (CVE-2006-0476), foi possível aceder à conta do utilizador que tinha o programa instalado.

4.6.3.2 Indicadores

Eventos OSSEC

- rule id=”40104” level="12": Possible buffer overflow attempt
- rule id=”40106” level="12": Buffer overflow attempt (probably on yppasswd)

Eventos SNORT

- COMMUNITY WEB-CLIENT Winamp PlayList buffer overflow attempt [100000228] classification: attempted administrator privilege gain [Priority: 1]

Rede

Total Bytes	7461 bytes
Total pacotes	38 pacotes
Tempo total	79.249 segundos
Pacotes/segundo	0.480 pac/seg
Tamanho médio dos pacotes	196 bytes
Bytes/segundo	94.147 bytes/seg
Protocolo	76.32% TCP 18.42% UDP 5.26% ICMP

Tabela 4.3 - Estatísticas de tráfego do Capec-100

4.6.4 Password Brute Forcing (CAPEC-49)

Esta técnica de ataque consiste em tentar todas as combinações possíveis de palavras passe, combinando todas as letras, números e caracteres.

Embora esta técnica não pareça por si só uma técnica que ofereça vantagem significativa ao atacante, devido à imensa quantidade de possibilidade, mostra-se na realidade muito útil. O utilizador utiliza muitas vezes palavras passe comuns e fracas, por comodidade. Estas podem ser adicionadas a dicionários de palavras passe, que servirão como base nos ataques de força bruta.

4.6.4.1 Técnica

No caso de estudo foi instalado o “microsoft sql server 2005” numa máquina, o nome de utilizador predefinido para acesso remoto à base de dados foi “sa”. Tendo em conta esse conhecimento e recorrendo a um dicionário de palavras passe mais utilizadas testaram-se possibilidades, até que surgiu uma que corresponde à palavra passe correta.

4.6.4.2 Indicadores

Eventos OSSEC

- rule id=”18180” level="5": MS SQL Server Logon Failure.
- rule id=”18152” level 10 : Multiple Windows Logon Failures

Eventos SNORT

- SQL sa brute force failed login unicode attempt (1:3273) classification: Unsuccessful User Privilege Gain, [Priority: 1]

Rede

Total Bytes	750035 bytes
Total pacotes	7467 pacotes
Tempo total	113.326 segundos
Pacotes/segundo	65.889 pac/seg
Tamanho médio dos pacotes	100 bytes
Bytes/segundo	6618.367 bytes/seg
Protocolo	99.99% TCP • 33.32% TDS

Tabela 4.4 - Estatísticas de tráfego do Capec-49

4.6.5 Restful Privilege Elevation (CAPEC-58)

Embora muitas vezes o atacante consiga ter acesso remoto à máquina, a conta à qual ele tem acesso não tem privilégios suficientes sobre a máquina alvo, não permitindo por exemplo a instalação de malware. Por esse motivo, torna-se necessário aumentar os privilégios do atacante para privilégios de administração sobre a máquina permitindo um maior controle sobre a mesma.

4.6.5.1 Técnica

Neste caso, após acesso à conta de um utilizador normal, foi utilizada a vulnerabilidade CVE-2010-0233, que permite o aumento de privilégios da conta de utilizador normal para conta de administração.

4.6.5.2 Indicadores

Eventos OSSEC

- rule id=”18153” level="10": Multiple audit failure events.

Eventos SNORT

Visto tratar-se de um evento local não gerando tráfego anormal na rede, não há eventos a registar a nível de rede.

Rede

Total Bytes	2434884 bytes
Total pacotes	25999 pacotes
Tempo total	210.002 segundos
Pacotes/segundo	12.376 pac/seg
Tamanho médio dos pacotes	936.854 bytes
Bytes/segundo	11594.577 bytes/seg
Protocolo	97.08% TCP

Tabela 4.5 - Estatísticas de tráfego do Capec-58

4.6.6 Men in the middle (CAPEC-94)

Este ataque tem como objetivo intercetar a comunicação entre duas máquinas de forma transparente para as duas máquinas alvo. Como todo o tráfego passa pelo atacante, este pode retirar toda a informação que não se encontre encriptada.

4.6.6.1 Técnica

Recorrendo a uma ferramenta disponibilizada no Kali Linux, de seu nome Ettercap, inunda-se a rede com pacotes ARP, de modo a que sempre que haja um pedido ARP o endereço MAC do atacante seja associado às máquinas alvo.

4.6.6.2 Indicadores

Eventos OSSEC

- rule id=”7200” level="0": MS SQL Server Logon Failure.
- rule id=”7202” level="9": The source mac Ethernet address didn't match the address inside the arp packet, probably arpspoofing attempt.

Eventos SNORT

De acordo com o manual do SNORT, recorrendo ao *ARPSpoof Preprocessor* disponível no sistema de deteção de intrusões, é possível ser alertado sempre que ocorrerem possíveis tentativas de *Men in The Middle*. Infelizmente na presente simulação não foi possível despoletar esse evento para verificar a sua importância no modelo. Após pesquisa detalhada na tentativa de solucionar o problema, chegou-se à conclusão que este era uma problema geral na comunidade Web e que o *ARPSpoof Preprocessor* deixou de funcionar nas versões mais recentes do SNORT.

Rede

Total Bytes	47713 bytes
Total pacotes	699 pacotes
Tempo total	212.209 segundos
Pacotes/segundo	3.294 pac/seg
Tamanho médio dos pacotes	68 bytes
Bytes/segundo	224.840
Protocolo	53.79% ARP 18.03% TCP 3.72% ICMP

Tabela 4.6 - Estatísticas de tráfego do Capec-94

4.6.7 Forceful Browsing (CAPEC-87)

O atacante tenta manipular as URL's de um *website*, de modo a tentar aceder a zonas restritas, através da alteração direta de parâmetros no URL.

4.6.7.1 Técnica

Através de uma simples *script* em Shell, que permite fazer *wget* a uma lista de diretorias previamente selecionadas, faz-se uma espécie de ataque de força bruta para verificar as diretorias existentes, analisando as que retornam o erro 404.

4.6.7.2 Indicadores

Eventos OSSEC

- rule id="31104" level="6": Common web attack.
- rule id="31511" level="6": Blacklisted user agent (wget)
- rule id="31153" level="10": Multiple common web attacks from same souce ip.

Eventos SNORT

Visto o tráfego viajar na rede não ser muito diferente de um tráfego normal não há eventos do SNORT a destacar, mas o número elevado de respostas com o erro "http número 404" é um sinal forte de *forcefull browsing*.

Rede

Total Bytes	797360 bytes
Total pacotes	6120 pacotes
Tempo total	49.546 Segundos
Pacotes/segundo	123.520 pac/seg
Tamanho médio dos pacotes	130 bytes
Bytes/segundo	16093.186 bytes/seg
Protocolo	100% TCP

Tabela 4.7 - Estatísticas de tráfego do Capec-87

4.6.8 SQL Injection (CAPEC-66)

Os erros de *SQL Injection* resultam em erros na validação do *input*, permitindo a injeção de pedidos à base de dados sem a devida validação. Este tipo de ataque permite o acesso a informação privilegiada, podendo permitir a alteração/destruição de dados na base de dados.

4.6.8.1 Técnica

Recorrendo à ferramenta SQLMAP, ferramenta disponível no sistema operativo KALI Linux, conseguiu-se de forma rápida e eficaz detetar falhas na validação. Estas falhas permitiam aceder indevidamente a todos os utilizadores e respetivas palavras-chave, da base de dados do servidor.

4.6.8.2 Indicadores

Eventos OSSEC

- rule id="31103" level="6": SQL injection attempt
- rule id="31152" level="10": Multiple SQL injection attempts from same souce ip

Eventos SNORT

Embora o SNORT contenha algumas regras pré-definidas para deteção de ataques de *SQL Injection* elas não são muito fiáveis. Isso deve-se ao facto dos ataques de *SQL Injection* funcionarem sobre a camada de aplicação, explorando problemas de validação.

Por não serem fiáveis os eventos do SNORT, não foram considerados neste caso.

Rede

Total Bytes	6984 bytes
Total pacotes	41 pacotes
Tempo total	20.991 segundos
Pacotes/segundo	1.953 pac/seg
Tamanho médio dos pacotes	170 bytes
Bytes/segundo	332.715 bytes/seg
Protocolo	63.41% TCP 29.27% ARP

Tabela 4.8 - Estatísticas de tráfego do Capec-66

4.6.9 HTTP DoS (CAPEC-469)

Este ataque tem como pressuposto o servidor atender os pedidos em *threads*. Criando várias *threads* que se apoderam de todos os recursos do servidor.

4.6.9.1 Técnica

Recorrendo a uma script em perl de nome "*slowloris*", é possível fazer um ataque inundando o servidor com pedidos http.

4.6.9.2 Indicadores

Eventos OSSEC

- rule id="31151" level="10": Multiple web server 400 error codes from same source ip.

Eventos SNORT

- Unusually Fast HTTP Attempt [1: 3000003:1] [Classification: Potential DOS] [Priority: 0]

Rede

Total Bytes	581307 bytes
Total pacotes	3785 pacotes
Tempo total	70.918 segundos
Pacotes/segundo	53.371 pac/seg
Tamanho médio dos pacotes	154 bytes
Bytes/segundo	8196.861 bytes/seg
Protocolo	99.68% TCP

Tabela 4.9 - Estatísticas de tráfego do Capec-469

4.6.10 TCP Flood (CAPEC-482)

Este ataque visa inundar a rede com pedidos transmission control protocol (TCP). Este tipo de ataque explora uma vulnerabilidade no protocolo TCP que obriga o servidor a manter a ligação em aberto.

4.6.10.1 Técnica

Recorrendo a uma script disponível no sistema operativo Kali Linux, de nome “hping”, é possível inundar o servidor com pacotes tcp, tornado o acesso ao servidor inviável.

4.6.10.2 Indicadores

Eventos OSSEC

Tal como indicado antes, este é um evento de rede, não deixando nenhum tipo vestígios no sistema.

Eventos SNORT

- BAD-TRAFFIC tcp port 0 traffic [1:524:8] Classification: Misc activity [Priority: 3]

Rede

Total Bytes	419146 bytes
Total pacotes	6980 pacotes
Tempo total	72.749 segundos
Pacotes/segundo	95.946 pac/seg
Tamanho médio dos pacotes	60 bytes
Bytes/segundo	5761.512 bytes/seg
Protocolo	99.01% TCP

Tabela 4.10 - Estatísticas de tráfego do Capec-482

4.6.11 Pointer Attack (CAPEC-129)

Técnica de ataque que manipula os apontadores de memória, direcionando-os para posições de memória indevidas. Este ataque pode resultar em falhas de sistema ou em injeção de código a ser executado.

4.6.11.1 Técnica

Recorrendo a uma vulnerabilidade conhecida nos Sistemas operativos da Microsoft (CVE-2012-0002), que contem um bug que permite executar código não autorizado através do protocolo de controlo remoto do Windows, conseguiu-se ativar um apontador de memória para um objeto inexistente, provocando a negação de serviço.

4.6.11.2 Indicadores

Eventos OSSEC

- rule id="18103" level="10": Windows error event.
- rule id="18105" level="4": Windows audit failure event

Eventos SNORT

- MISC MS Terminal server request [1:1448:12] Classification: Generic Protocol Command Decode [Priority: 3]

Rede

Total Bytes	9123 bytes
Total pacotes	78 pacotes
Tempo total	37.672 segundos
Pacotes/segundo	2.071 pac/seg
Tamanho médio dos pacotes	117 bytes
Bytes/segundo	242.171 bytes/seg
Protocolo	51.28% UDP 46.15% TCP

Tabela 4.11 - Estatísticas de tráfego do Capec-129

4.7 Eventos relevantes

Após a escolha e adaptação do modelo adequado são apresentados todos os eventos recolhidos mas já aplicados na fase de decisão apropriada. Por exemplo, os eventos gerados durante um ataque de *SQL Injection* devem ser colocados sob a fase de decisão de *Compromise Data Integrity*. Deste modo é possível ter uma noção dos eventos gerados em cada fase de decisão.

- R: Reconnaissance :
  - Eventos OSSEC
  - Eventos SNORT
    - ICMP PING NMAP [1:469:1] [Classification: Attempted Information Leak] [Priority: 2]

  - Rede

- GA: Gain Access :
    - Eventos OSSEC
        - rule id=”40104” level="12": Possible buffer overflow attempt
        - rule id=”40106” level="12": Buffer overflow attempt (probably on yppasswd)
        - rule id=”18180” level="5": MS SQL Server Logon Failure.
        - rule id=”18152” level 10 : Multiple Windows Logon Failures
    - Eventos SNORT
        - COMMUNITY WEB-CLIENT Winamp PlayList buffer overflow attempt [100000228] classification: attempted administrator privilege gain [Priority: 1]
        - SQL sa brute force failed login unicode attempt (1:3273) classification: Unsuccessful User Privilege Gain, [Priority: 1]
    - Rede
- PE: Privilege escalation:
    - Eventos OSSEC
        - rule id=”18153” level="10": Multiple audit failure events.
    - Eventos SNORT
    - Rede
- VB: Victim Browsing :
    - Eventos OSSEC
        - rule id=”7200” level="0": MS SQL Server Logon Failure.
        - rule id=”7202” level="9": The source mac Ethernet address didn't match the address inside the arp packet, probably arpspoofing attempt.
        - rule id="31104" level="6": Common web attack.
        - rule id="31511" level="6": Blacklisted user agent (wget)

        - rule id="31153" level="10": Multiple common web attacks from same souce ip.

    - Eventos SNORT
    - Rede

- HT: Hide Traces:
    - Eventos OSSEC
    - Eventos SNORT
    - Rede

- EP: Execute Program:
    - Eventos OSSEC
    - Eventos SNORT
    - Rede

- IMC: Implant Malicious Code :
    - Eventos OSSEC
    - Eventos SNORT
    - Rede

- CDI: Compromise Data Integrity:
    - Eventos OSSEC
        - rule id="31103" level="6": SQL injection attempt
        - rule id="31152" level="10": Multiple SQL injection attempts from same souce ip

    - Eventos SNORT
    - Rede

- DoS: Denial of Service:
    - Eventos OSSEC
        - rule id="31151" level="10": Multiple web server 400 error codes from same source ip.
        - rule id="18103" level="10": Windows error event.
        - rule id="18105" level="4": Windows audit failure event

    - Eventos SNORT
        - Unusually Fast HTTP Attempt [1: 3000003:1] [Classification: Potential DOS] [Priority: 0]
        - BAD-TRAFFIC tcp port 0 traffic [1:524:8] Classification: Misc activity [Priority: 3]
        - MISC MS Terminal server request [1:1448:12] Classification: Generic Protocol Command Decode [Priority: 3]

    - Rede

4.8 Software de injeção de tráfego malicioso

Uma grande dificuldade no que toca à avaliação de sistemas de deteção de intrusões prende-se com a criação de tráfego malicioso na rede capaz de gerar eventos nos sistemas de deteção de intrusões. Esta dificuldade de avaliação debate-se com a falta de conhecimento ao nível de ferramentas de penetração ou quando envolve testes de *stress* nas redes de empresas, por temor das mesmas relativamente à utilização de ferramentas adequadas para ataques, mesmo que utilizadas de forma responsável.

A necessidade desta ferramenta surgiu pela grande quantidade de tempo despendido no estudo de técnicas ofensivas, no decorrer deste projeto. Este conhecimento era imperioso para a captura dos eventos, para o modelo desenvolvido.

Por este motivo, a criação de um *software* capaz de reproduzir o tráfego malicioso capturado previamente, mostra-se como uma mais-valia, como ferramenta de apoio para testes de sistemas de deteção de intrusões. Exemplificando, um investigador que pretenda fazer um teste (ataque) a um sistema de deteção de intrusões sem o conhecimento de como o ataque é feito, apenas necessita de selecionar a vítima e o ficheiro pcap com o tráfego do ataque que desejar testar.

Para a criação desse *software*, como explanado no tópico 4.1, é necessário abrigara compilação de uma extensa biblioteca de ficheiro PCAP, que permita a reintrodução dos pacotes que constam nesse ficheiro novamente na rede, despoletando os eventos nos sistemas de deteção de intrusões. Essa mesma biblioteca deverá estar organizada primeiramente de acordo com as fases do modelo proposto e posteriormente de acordo com a lista de ataques disponibilizada na biblioteca de ataques CAPEC. Desse modo, sempre que um investigador pretenda testar um ataque específico dessa biblioteca, poderá recorrer à ferramenta desenvolvida. É de notar que na descrição oferecida pelo CAPEC não há apoio em como realizar o ataque.

Como requisito elementar do *software* criado, pretende-se que este seja capaz de alterar o endereço IP de origem e destino assim como o endereço media access control

(MAC) dos datagramas, recalculando no final o *checksum* dos mesmo, como forma a evitar erros de integridade no datagrama.

Python foi a opção de linguagem de programação escolhida para a criação do *software*. Esta opção prendeu-se com o facto da existência da biblioteca SCAPY[17], capaz de oferecer funções que lidam com o todo o tipo de tarefas de rede, desde a substituição de componentes do pacote até à injeção de tráfego na camada 2.

4.8.1 Fluxograma

O software desenvolvido foi baseado no fluxograma apresentado na Figura 4.4.

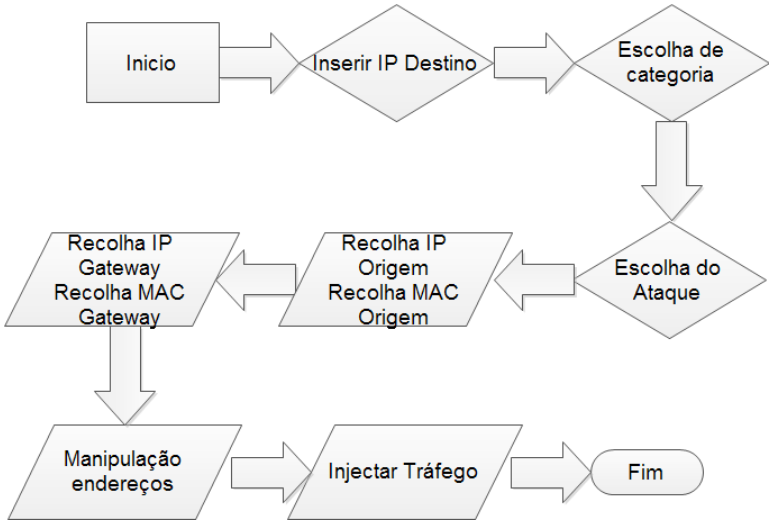


Figura 4.4 – Fluxograma do Software

Os únicos dados requeridos aos utilizadores do *software*, para realizar o ataque serão: a categoria do ataque a realizar, escolha do ataque de acordo com a biblioteca de ataques CAPEC e o IP da vítima. Assumindo que a máquina que está a executar o *software* é o atacante, o software é capaz de recolher o seu IP, endereço MAC assim

[17] http://www.secdev.org/projects/scapy/

como o IP e endereço MAC do *gateway*, o que permite realizar ataques externos à rede em que se encontra o atacante.

4.8.2 Algoritmo

O algoritmo seguinte representa, em linguagem informal, a sequência do software desenvolvido para injeção de tráfego malicioso na rede, proveniente do *dataset* recolhido, o código original pode ser consultado no Apêndice D.

1. Listar interfaces
2. Escolher Interfaces
3. Inserir IP da Vitima
4. Escolher classe do ataque
5. Escolher ataque
6. Receber IP atacante
7. Receber Endereço MAC atacante
8. Receber IP *Gateway*
9. Receber Endereço MAC *Gateway*
10. Ler ficheiro capec-X.pcap
11. Enquanto ficheiro != fim
12. Se datagrama proveniente atacante
    i. Pkt.src[IP]= IP atacante
    ii. Pkt.src[Ether]= Endereço MAC atacante
    iii. Pkt.dst[IP]= IP vitima
    iv. Pkt.dst[Ether]= Endereço MAC Gateway
13. Se não
    i. Pkt.src[IP]= IP vitima
    ii. Pkt.src[Ether]= Endereço MAC vitima
    iii. Pkt.dst[IP]= IP atacante
    iv. Pkt.dst[Ether]= Endereço MAC

14. Limpar *checksum* do datagrama
15. Recalcular novo *checksum*
16. Injetar datagrama na camada 2

4.8.3 Teste do *software* de injeção de tráfego malicioso

Tal como referido previamente, um *software* que recorra a uma biblioteca de ataques criada previamente a partir de ficheiros pcap, deve ser capaz de alterar 3 componentes essenciais de um datagrama. Esses componentes são: o endereço IP, endereço MAC tanto referente à vítima como ao atacante e o *checksum* do datagrama, de forma a evitar problemas na verificação da integridade. Estes requisitos são imprescindíveis para a simulação em qualquer máquina ou rede independentemente da topologia, como referido nos requisitos do modelo.

Na Figura 4.5 pode ser visto um datagrama original pertencente ao *dataset*. Na Figura 4.6 pode ser visto, o mesmo datagrama com novos *IP's* e novos endereço MAC.

Figura 4.5 - Exemplo de datagrama original

Figura 4.6 – Exemplo de datagrama modificado

É possível adicionalmente, verificar que o *checksum* do datagrama modificado se encontrar vazio, pois o novo *checksum* é calculado pela função que injeta o tráfego na camada 2. Tal não é passível de ser visualizado no terminal mas pode ser confirmado no software *Wireshark,* como é possível observar na Figura 4.7.

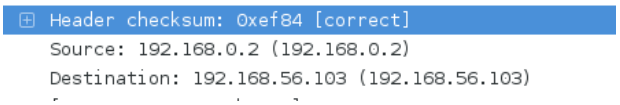


Figura 4.7 - Verificação Wireshark

4.8.4 Interface gráfica

A interface gráfica do ataque deve ser simples e intuitiva, facilitando a integração dos gestores de rede com a ferramenta.

Por forma a facilitar a utilização os dados necessários para alteração dos datagramas, estes são recolhidos de forma automatizada, restando ao investigador a escolha do ataque a realizar, tendo conhecimento prévio do IP da vítima. Um exemplo pode ser visto na Figura 4.8.

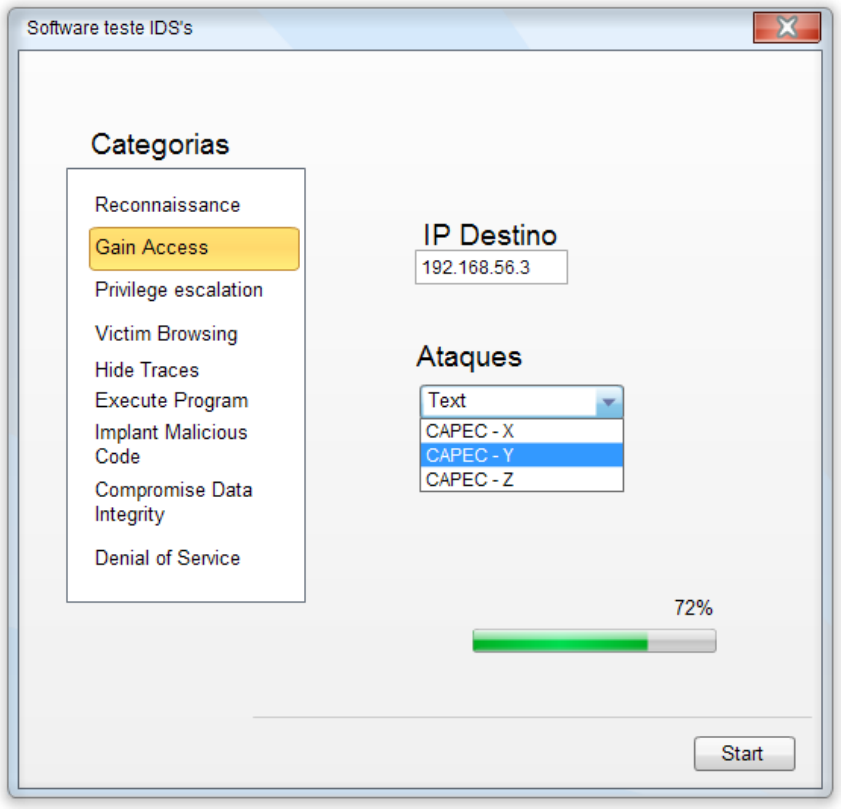


Figura 4.8 – Possível interface gráfica do software

Capítulo 5

5 Conclusões

No presente capítulo serão apresentados as conclusões sobre os objetivos atingidos, a par de uma breve análise e descrição acerca das principais dificuldades do desenvolvimento deste projeto.

Posteriormente será apresentada uma breve reflexão sobre novos caminhos que este projeto permitiu abrir e o quanto ainda pode ser feito com o propósito de melhorar a solução aqui proposta.

5.1 Conclusões

No projeto desenvolvido foi proposto um modelo comportamental de ataques em redes informáticas. Este modelo foi criado com o propósito de fornecer apoio na avaliação dos sistemas de deteção de intrusões.

Foi realizado um estudo aprofundado para compreensão dos principais objetivos e necessidades na avaliação de sistemas de deteção de intrusões, para melhor compreensão acerca das vantagens da criação de um modelo de ataques. Para além do referido, foram estudados os modelos de ataques existentes, assim como os seus modos de representação mais relevantes, de modo a projetar um modelo de ataques adequado aos requisitos propostos. Ainda com esse intuito, foram estudados três soluções de modelos de já existentes, a par das bibliotecas de ataques.

No ponto seguinte, foi desenvolvido um modelo de ataques que cumprisse os requisitos proposto. Como resultado desse estudo foi desenvolvido um modelo capaz de abranger todos os tipos de ataques (automatizado ou não), independentemente da

topologia. Este modelo foi desenvolvido para ser de fácil automatização e capaz de distinguir de forma fácil eventos/ações de transições.

Com a criação do modelo de ataques foram estabelecidas nove fases de decisão, que visam de forma atómica descrever um ponto no tempo de um ataque, mas que quando ligadas entre si permitem descrever o ataque como um todo. De seguida foi premente criar um ambiente controlado para realização do ataque, recolhendo deste modo eventos gerados em cada fase de decisão. Adicionalmente foi recolhido o tráfego de rede, para o desenvolvimento de uma ferramenta de injeção de tráfego malicioso.

Por fim e como forma a colmatar uma das maiores dificuldades sentidas, foi desenvolvido um *software* que com base no *dataset* recolhido durante a simulação. Este *software* mostrou-se capaz de alterar componentes dos datagrama e injetar os mesmos na rede, permitindo fazer testes de penetração sem um conhecimento aprofundado das ferramentas fundamentais para o efeito.

Em conclusão, o modelo de ataques desenvolvido revelou imensas vantagens e possibilidades de implementação. O modelo mostrou-se capaz de abranger todos os ataques, recolher eventos provenientes de três fontes distintas (NIDS, HIDS e estatísticas de rede) gerando eventos sem precedentes. Isto permite avaliar um vasto leque de sistemas de deteção de intrusões independentemente da topologia de rede, facilitando a integração e facilidade de utilização. Adicionalmente, este modelo é passível de automatização, sendo que uma parte significativa desse *software* já se encontra desenvolvido e exposto na presente dissertação.

5.2 Dificuldades

A principal dificuldade sentida no que toca a avaliação de sistemas de deteção de intrusões prende-se com a criação de ataques específicos para verificação dos eventos gerados. Embora as bibliotecas de ataques, como o CAPEC, estejam bastante completas ao nível das vulnerabilidades, recursos atingidos durante o ataque e até

mesmo descrição teórica do funcionamento, faltava algo que clarifique como explorar essas vulnerabilidades descritas. Nesta lacuna foi despendido grande parte do tempo da investigação, estudando técnicas ofensivas na área de segurança de informação.

Devido à limitação de recursos, não foi possível efetuar testes com ataques de DDoS. Para tal seria necessário construir uma *botnet*, o que com os recursos disponíveis se mostrou inviável.

Referente às estatísticas de rede recolhidas como possível apoio aos indicadores também se tornou uma dificuldade a escassez de tempo para realizar o mesmo ataque várias vezes recolhendo uma quantidade maior de dados estatísticos que possam ser trabalhados.

5.3 Trabalho Futuro

O modelo de ataques proposto revela-se bastante robusto e sólido, não sendo necessário alterações na sua estrutura, de momento.

É de notar, no entanto que a sua base de conhecimento de ataques, contem apenas onze ataques testados em ambiente controlado, devido ao facto de cada ataque novo adicionado requerer o dispêndio de muito tempo. Como mais-valia para o modelo de ataques e software de injeção de tráfego malicioso seria importante aumentar essa base de conhecimento, testando para recolha de eventos e *datasets* novos ataques que se encontram no Apêndice A.

Pode ser também uma mais-valia para o projeto a realização variada do mesmo tipo de ataque, utilizando outras máquinas ou explorando outras vulnerabilidades, com o propósito de recolher um maior e mais variado tráfego de permitindo assim detetar padrões estatísticos no tráfego recolhido.

Por não ser objetivo inicial deste projeto, a componente de *software* de injeção de tráfego malicioso poderia ter sido mais desenvolvido, o que não aconteceu por falta de tempo. No *software* desenvolvido, é necessário a conclusão da componente de

encaminhamento, permitindo que o próprio *software* seja capaz de desenhar a tabela de encaminhamento, quando o sistema operativo ainda não o realizou. Para além disto, falta criar uma interface gráfica, como sugerida no subtópico 4.8.4.

Bibliografia

6 Bibliografia

1. Cheung, S., Fong, M. W., Ave, R., & Park, M. (2003). Modeling Multistep Cyber Attacks for Scenario Recognition ∗, I(Discex Iii), 284–292.
2. Daley, K., Larson, R., & Dawkins, J. (2002). A structural framework for modeling multi-stage network attacks. Proceedings. International Conference on Parallel Processing Workshop, 5–10. doi:10.1109/ICPPW.2002.1039705
3. Gadelrab, M., Kalam, A. A. El, & Deswarte, Y. (2008). Execution Patterns in Automatic Malware and Human-Centric Attacks. 2008 Seventh IEEE International Symposium on Network Computing and Applications, 29–36. doi:10.1109/NCA.2008.37
4. Interoperability, I. D. (n.d.). InfoSec Reading Room tu , A ho ll r igh.
5. Kotenko, I., & Chechulin, A. (2013). A Cyber Attack Modeling and Impact Assessment framework. Cyber Conflict (CyCon), 2013 5th
6. Kuhl, M. E. (2007). Proceedings of the 2007 Winter Simulation Conference S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds., 1180–1188.
7. Lathrop, S., Hill, J., & Surdu, J. (2003). Modeling network attacks on Behavior Representation in Modeling
8. Mirembe, D. P., & Muyeba, M. (2008). Threat Modeling Revisited: Improving Expressiveness of Attack. 2008 Second UKSIM European Symposium on Computer Modeling and Simulation, 93–98. doi:10.1109/EMS.2008.83
9. Mahoney, M. V., & Chan, P. K. (2002). Learning nonstationary models of normal network traffic for detecting novel attacks. Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD ’02, 376. doi:10.1145/775094.775102
10. Saber, M., & Bouchentouf, T. (2010). Generation of attack scenarios by modeling algorithms for evaluating IDS.

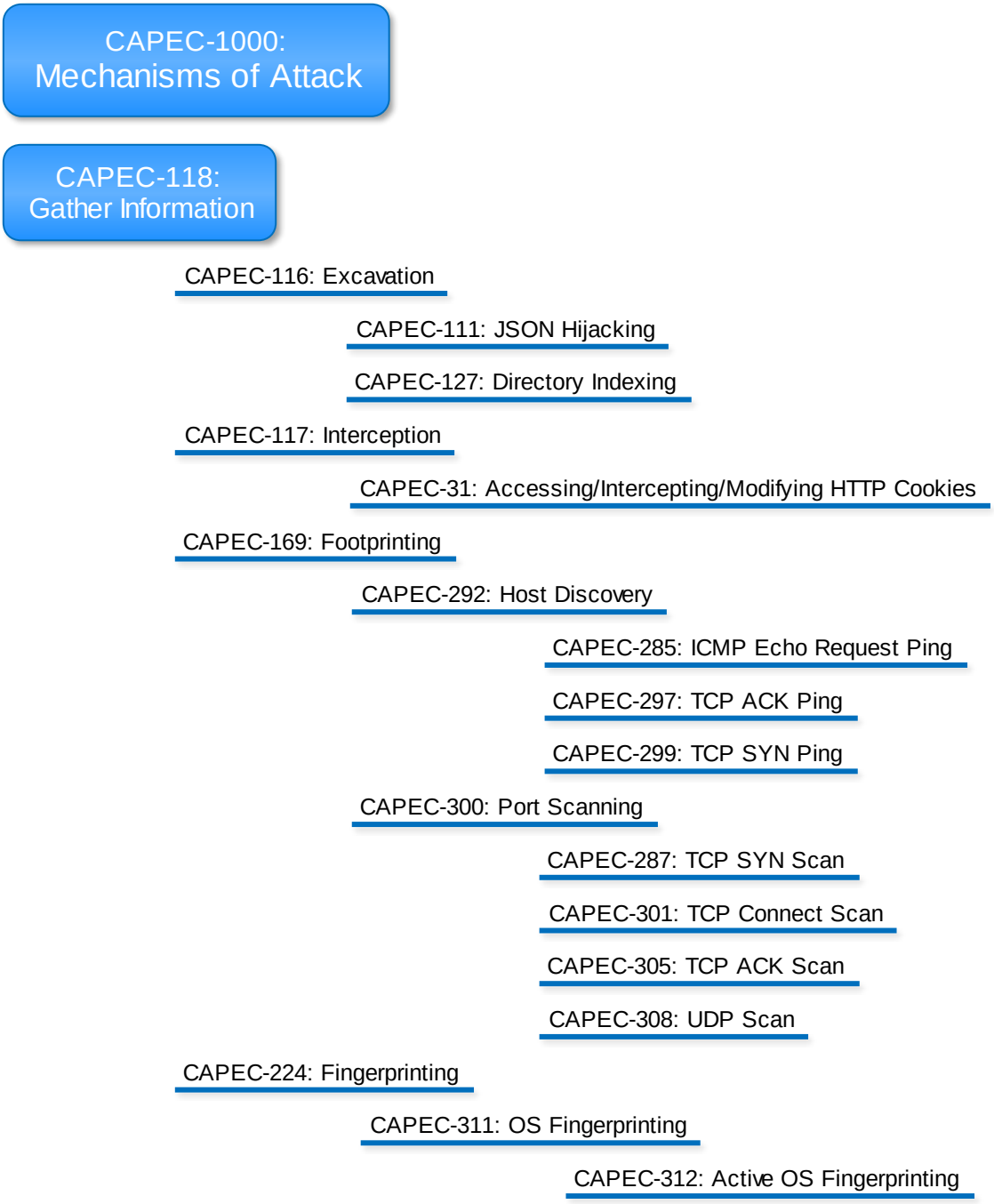
11. Saini, V., Duan, Q., & Paruchuri, V. (2008). Threat modeling using attack trees. Journal of Computing Sciences in …, 124–131. Retrieved from http://dl.acm.org/citation.cfm?id=1352100
12. Templeton, S. J., & Levitt, K. (2000). A requires/provides model for computer attacks. Proceedings of the 2000 Workshop on New Security Paradigms - NSPW '00, 31–38. doi:10.1145/366173.366187
13. Morgenstern, M., & Pilz, H. (2010). Useful and useless statistics about viruses and anti-virus programs. Proceedings of the CARO Workshop.
14. Shostack, A. (2014). Threat Modeling: Designing for Security. Vasa. Retrieved from http://threatmodelingbook.com/index.html
15. Scarfone, K., & Mell, P. (n.d.). Guide to Intrusion Detection and Prevention Systems (IDPS) Recommendations of the National Institute of Standards and Technology.
16. Gengler, C. E., Rossi, M., Hui, W., & Bragge, J. (2006). THE DESIGN SCIENCE RESEARCH PROCESS : A MODEL FOR PRODUCING AND PRESENTING INFORMATION.
17. Edition, F. (2012). Principles of Information Security Fourth Edition. (H. J. Mattord & M. E. Whitman, Eds.) (Fourth Edi.).
18. Charlie Scott, Paul Wolfe, and B. H. (Ed.). (2004). Snort for Dummies. Wiley Publishing, Inc.
19. Burroughs, J., & Engebretson, P. (2010). Attack Traffic Libraries for Testing and Teaching Intrusion Detection Systems. Dsu.edu.
20. Chappell, L. (Ed.). (2013). Wireshark 101 (1st Editio.). Chappell University.
21. Wilhelm, T. (Ed.). (2013). Professional Penetration Testing (Second Edi.). Elsevier.
22. Vulnerable, D., & Application, W. (2010). Damn Vulnerable Web Application (DVWA) Official Documentation, 1–16.
23. Kennedy, D., Gorman, J. O., Kearns, D., & Aharoni, M. (2011). Metasploit The Penetration Tester's Guide. William Pollock.
24. Dahl, O. M., & Wolthusen, S. D. (n.d.). Modeling and Execution of Complex Attack Scenarios using Interval Timed Colored Petri Nets. Fourth IEEE

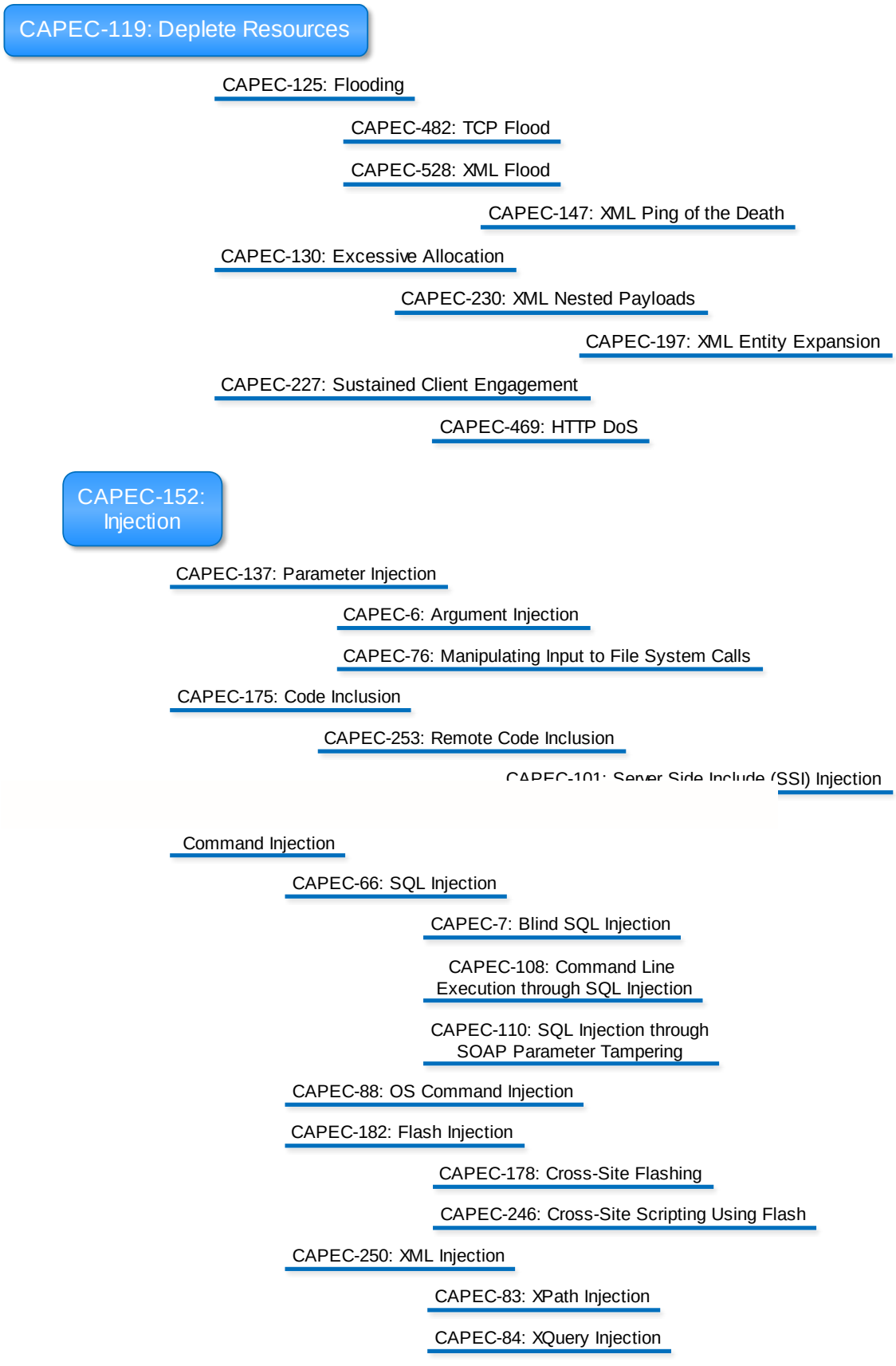
International Workshop on Information Assurance (IWIA'06), 157–168. doi:10.1109/IWIA.2006.17

25. Engebretson, P., Cronin, K., & Pauli, J. (2010). SprayPAL : How capturing and replaying attack traffic can save your IDS / IPS 2 . Building a . pcap Attack Library (PAL), 1–3.
26. CLURE, S. M. C., SCAMBRAY, J., & KURTZ, G. (2009). HACKING EXPOSED ™ 6: NETWORK SECURITY SECRETS & SOLUTIONS. The McGraw-Hill Companies.
27. MELO, S. DE. (2004). FERRAMENTAS DE SEGURANÇA: CRIMES OCORRIDOS NA REDE. Unipac.br. Retrieved from http://ftp.unipac.br/site/bb/tcc/tcc-914eec7159485f8900545e9467762a7c.pdf
28. García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 28(1-2), 18–28. doi:10.1016/j.cose.2008.08.003
29. Tjhai, G., & Papadaki, M. (2008). Investigating the problem of IDS false alarms: An experimental study using Snort. … of the IFIP TC 11 23rd …, 278, 253–267. Retrieved from http://link.springer.com/content/pdf/10.1007/978-0-387-09699-5_17.pdf
30. Pietraszek, T. (2004). Using adaptive alert classification to reduce false positives in intrusion detection. Recent Advances in Intrusion Detection, 1–24. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-30143-1_6
31. Alhomoud, A., Munir, R., Disso, J. P., Awan, I., & Al-Dhelaan, a. (2011). Performance Evaluation Study of Intrusion Detection Systems. Procedia Computer Science, 5, 173–180. doi:10.1016/j.procs.2011.07.024
32. Spathoulas, G. P., & Katsikas, S. K. (2010). Reducing false positives in intrusion detection systems. Computers & Security, 29(1), 35–44. doi:10.1016/j.cose.2009.07.008
33. http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/

Apêndice A

A. Lista hierárquica de ataques escolhidos




CAPEC-119: Deplete Resources
CAPEC-125: Flooding
CAPEC-482: TCP Flood
CAPEC-528: XML Flood
CAPEC-147: XML Ping of the Death
CAPEC-130: Excessive Allocation
CAPEC-230: XML Nested Payloads
CAPEC-197: XML Entity Expansion
CAPEC-227: Sustained Client Engagement
CAPEC-469: HTTP DoS
CAPEC-152: Injection
CAPEC-137: Parameter Injection
CAPEC-6: Argument Injection
CAPEC-76: Manipulating Input to File System Calls
CAPEC-175: Code Inclusion
CAPEC-253: Remote Code Inclusion
CAPEC-101: Server Side Include (SSI) Injection
Command Injection
CAPEC-66: SQL Injection
CAPEC-7: Blind SQL Injection
CAPEC-108: Command Line Execution through SQL Injection
CAPEC-110: SQL Injection through SOAP Parameter Tampering
CAPEC-88: OS Command Injection
CAPEC-182: Flash Injection
CAPEC-178: Cross-Site Flashing
CAPEC-246: Cross-Site Scripting Using Flash
CAPEC-250: XML Injection
CAPEC-83: XPath Injection
CAPEC-84: XQuery Injection

CAPEC-210: Abuse of Functionality

CAPEC-212: Functionality Misuse

CAPEC-2: Inducing Account Lockout

CAPEC-50: Password Recovery Exploitation

CAPEC-223: Probabilistic Techniques - 223

CAPEC-112: Brute Force

CAPEC-49: Password Brute Forcing

CAPEC-16: Dictionary-based Password Attack

CAPEC-225: Exploitation of Authentication

CAPEC-128: Integer Attacks

CAPEC-92: Forced Integer Overflow

CAPEC-232: Exploitation of Authorization

CAPEC-22: Exploiting Trust in Client

CAPEC-94: Man in the Middle Attack

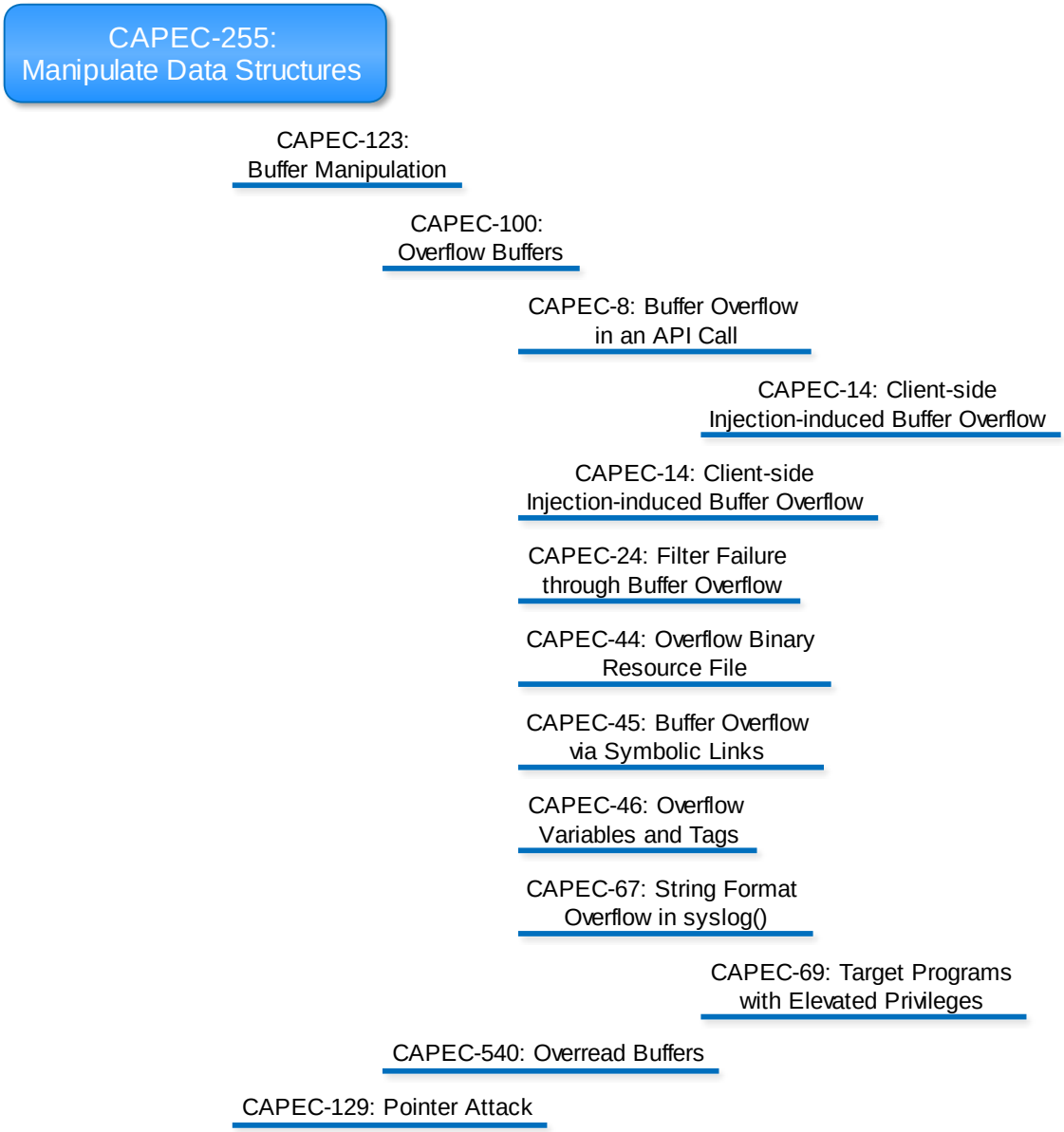
CAPEC-30: Hijacking a Privileged Thread of Execution

CAPEC-236: Catching exception throw/signal from privileged block

CAPEC-233 Privilege Escalation

CAPEC-17: Accessing, Modifying or Executing Executable Files

CAPEC-58: Restful Privilege Elevation


CAPEC-255:
Manipulate Data Structures
CAPEC-123:
Buffer Manipulation
CAPEC-100:
Overflow Buffers
CAPEC-8: Buffer Overflow
in an API Call
CAPEC-14: Client-side
Injection-induced Buffer Overflow
CAPEC-14: Client-side
Injection-induced Buffer Overflow
CAPEC-24: Filter Failure
through Buffer Overflow
CAPEC-44: Overflow Binary
Resource File
CAPEC-45: Buffer Overflow
via Symbolic Links
CAPEC-46: Overflow
Variables and Tags
CAPEC-67: String Format
Overflow in syslog()
CAPEC-69: Target Programs
with Elevated Privileges
CAPEC-540: Overread Buffers
CAPEC-129: Pointer Attack

Apêndice B

B. Diagrama de Ação

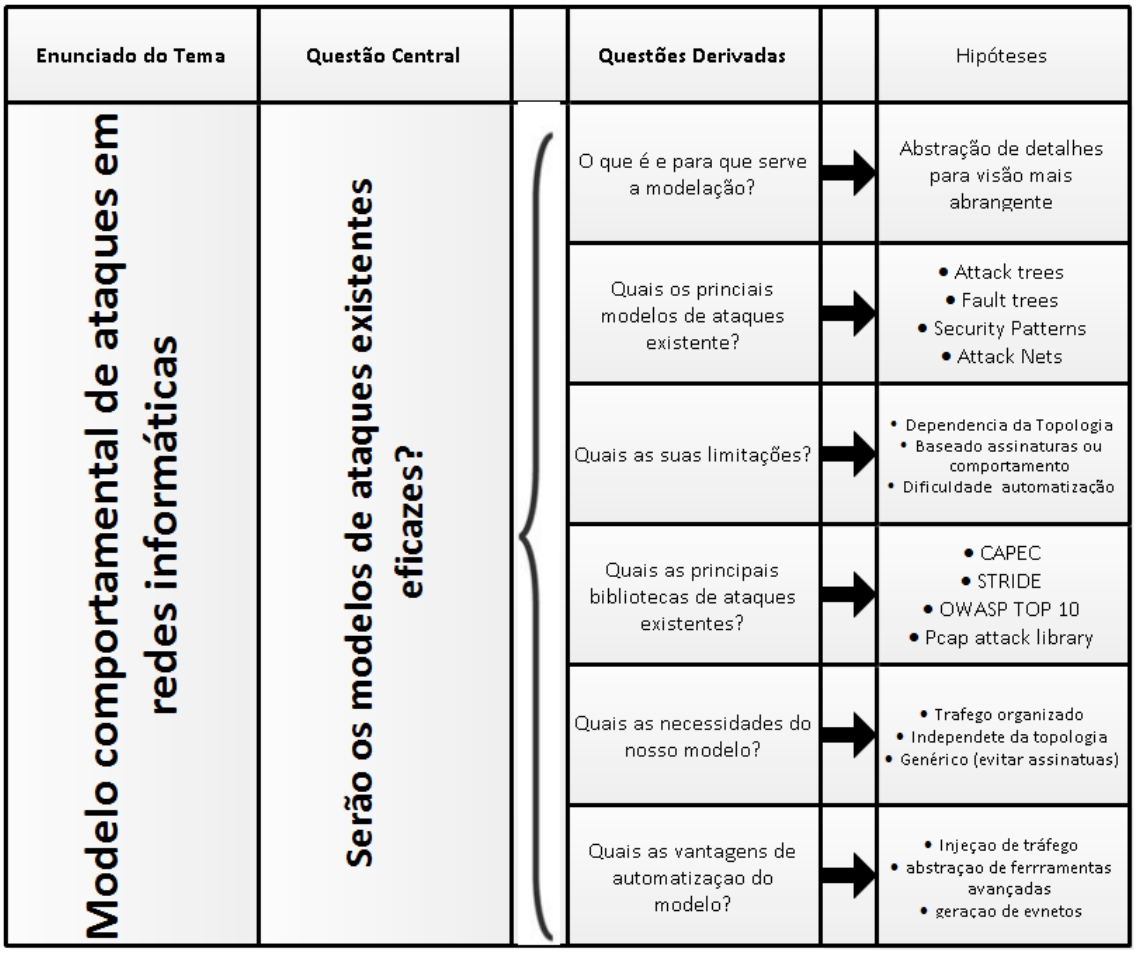


Figura B.1 – Diagrama de Ação

Apêndice C

C. Exemplo Ataque CAPEC

Common Attack Pattern Enumeration and Classification
A Community Resource for Identifying and Understanding Attacks

About CAPEC
Documents
Glossary
FAQs
CAPEC List
Search
Review
Downloads
Documentation
Release Notes
Archive
Submit Content
Community
Use & Citations
Related Activities
Discussion List
Contact Us
Compatibility
Program
Requirements
Participants
Make a Declaration
News & Events
Calendar
Free Newsletter
Search the Site

Attack Pattern ID: 89
Abstraction: Standard
Status: Draft
Completeness: Complete

Description

Summary

A pharming attack occurs when the victim is fooled into entering sensitive data into supposedly trusted locations, such as an online bank site or a trading platform. An attacker can impersonate these supposedly trusted sites and have the victim be directed to his site rather than the originally intended one.

Pharming does not require script injection or clicking on malicious links for the attack to succeed.

Attack Execution Flow

Exploit

1. Attacker sets up a system mocking the one trusted by the users. This is usually a website that requires or handles sensitive information.
2. The attacker then poisons the resolver for the targeted site. This is achieved by poisoning the DNS server, or the local hosts file, that directs the user to the original website
3. When the victim requests the URL for the site, the poisoned records direct the victim to the attackers' system rather than the original one.
4. Because of the identical nature of the original site and the attacker controlled one, and the fact that the URL is still the original one, the victim trusts the website reached and the attacker can now "farm" sensitive information such as credentials or account numbers.

Attack Prerequisites

- Vulnerable DNS software or improperly protected hosts file or router that can be poisoned
- A website that handles sensitive information but does not use a secure connection and a certificate that is valid is also prone to pharming

Typical Severity

Very High

Typical Likelihood of Exploit

Likelihood: High

Methods of Attack

- Spoofing
- Analysis
- Modification of Resources

Examples-Instances

Description

An online bank website requires users to provide their customer ID and password to log on, but does not use a secure connection.

An attacker can setup a similar fake site and leverage pharming to collect this information from unknowing victims.

Attacker Skills or Knowledge Required

Skill or Knowledge Level: Medium

The attacker needs to be able to poison the resolver - DNS entries or local hosts file or router entry pointing to a trusted DNS server - in order to successfully carry out a pharming attack. Setting up a fake website, identical to the targeted one, does not require special skills.

Resources Required

Except having enough knowledge of the way the targeted site has been structured in order to create a fake version, no additional resources are required. Poisoning the resolver requires knowledge of a vulnerability that can be exploited.

Probing Techniques

Description

The attacker observes the targeted website for use of secure connection to exchange sensitive information. If it does not use secure connections, victim users cannot distinguish between the original and fake versions of the website.

Description

The attacker can also fingerprint the software running on the targeted system (DNS server, router or host) and look for vulnerabilities in order to poison the entries.

Solutions and Mitigations

All sensitive information must be handled over a secure connection.

Known vulnerabilities in DNS or router software or in operating systems must be patched as soon as a fix has been released and tested.

End users must ensure that they provide sensitive information only to websites that they trust, over a secure connection with a valid certificate issued by a well-known certificate authority.

Attack Motivation-Consequences

Scope	Technical Impact	Note
Confidentiality	Read application data	

Related Weaknesses

CWE-ID	Weakness Name	Weakness Relationship Type
346	Origin Validation Error	Targeted
247	DEPRECATED (Duplicate): Reliance on DNS Lookups in a Security Decision	Targeted
292	DEPRECATED (Duplicate): Trusting Self-reported DNS Name	Targeted

Related Vulnerabilities

Vulnerability ID	Relationship Description
CVE-2005-0877	Dnsmasq before 2.21 allows remote attackers to poison the DNS cache via answers to queries that were not made by Dnsmasq.
CVE-2004-1754	The DNS proxy (DNSd) for multiple Symantec Gateway Security products allows remote attackers to poison the DNS cache via a malicious DNS server query response that contains authoritative or additional records.

Related Attack Patterns

Nature	Type	ID	Name	
ChildOf	A	151	Identity Spoofing	1000
ChildOf	A	161	Infrastructure Manipulation	1000
CanFollow	A	141	Cache Poisoning	1000
CanFollow	A	142	DNS Cache Poisoning	1000

Related Security Principles

- Reluctance To Trust
- Promoting Privacy

Related Guidelines

- Use Authentication Mechanisms, Where Appropriate, Correctly
- Use Well-Known Cryptography Appropriately and Correctly

Purposes

- Reconnaissance

CIA Impact

Confidentiality Impact:	Integrity Impact:	Availability Impact:
High	High	Low

Technical Context

Architectural Paradigms	Client-Server SOA
Frameworks	All
Platforms	All
Languages	All

Content History

Submitter	Organization	Date	Source
CAPEC Content Team	The MITRE Corporation	2014-06-23	Internal_CAPEC_Team

Apêndice D

D. Código do *software* de injeção de tráfego malicioso

```
import logging
logging.getLogger("scapy.runtime").setLevel(loggi
ng.ERROR)
from scapy.all import *
from scapy.utils import *
from subprocess import Popen, PIPE
import netifaces
contador=0


# Lista das interfaces disponiveis
print "Interfaces disponiveis:"
print netifaces.interfaces()
print "\n\n"
#interface = '\"eth1\"'


#Dados do atacante
addrs = netifaces.ifaddresses("eth1")
obj1 = addrs[netifaces.AF_INET]
obj2 = addrs[netifaces.AF_LINK]
att_ip = obj1[0]['addr']
att_mac = obj2[0]['addr']
print "IP do atacante:"
print att_ip
print "MAC do atacante:"
print att_mac
```

```
# Obter o IP do gateway
obj3= netifaces.gateways();
att_gateway_ip =
obj3['default'][netifaces.AF_INET][0]
print "IP do gateway"
print att_gateway_ip
#Obter o MAC do gatway
pid = Popen(["arp", "-n", att_gateway_ip],
stdout=PIPE)
s = pid.communicate()[0]
vic_mac = re.search(r"(([a-f\d]{1,2}\:){5}[a-
f\d]{1,2})", s).groups()[0]
print "MAC do Gateway do atacante:"
print vic_mac


#Dados da vitima
vic_ip = '192.168.56.105'
print "IP da vitima"
print vic_ip

#Escolha o ficheiro pcap
file = "capec49.pcap"

try:
     pkts=rdpcap(file)  # rdpcap("filename",X)
mostra apenas os X primeiros pacotes

     #Deterina quem e o atacante
     orig_ip_addr=pkts[0][IP].src

     print "Endereco origem:"
     print orig_ip_addr
```

```
        for pkt in pkts:
            print "\n\n\n\n"
            if pkt[IP].src == orig_ip_addr:
                print "sou o atacante"
                print "pacote original:"
                print pkt.command()
                pkt[IP].src= att_ip
                pkt[IP].dst= vic_ip
                pkt[Ether].src = att_mac
                pkt[Ether].dst = vic_mac
            else:
                print "sou a vitima"
                print "pacote original:"
                print pkt.command()
                pkt[IP].src= vic_ip
                pkt[IP].dst= att_ip
                pkt[Ether].src = vic_mac
                pkt[Ether].dst = att_mac


            print "pacote modificado:"
            print pkt.command()
            print "\n\n\n\n"
            del(pkt.chksum)
            sendp(pkt)
            #Contador de pacotes
            contador=contador+1
            print “Numero de pacotes enviados:”
            print contador

    except IOError:
        #Caso nao consiga ler ficheiro
        print "erro de leitura" %file
```

Apêndice E

E. Tabela comparativa de tráfego recolhido

	Trefego Normal	CAPEC-169	CAPEC-312	CAPEC-49	CAPEC-100	CAPEC-58	CAPEC-94
Total Bytes	1035141538	60322,98	316177	750035	7461	2434884	47713
Total pacotes	1164651	860	4533	7467	38	25999	699
Tempo total	5204,921	34,472	110,762	113,326	79,249	210,002	212,209
Pacotes/segundo	223,76	24,947	28,625	65,889	0,48	12,376	3,294
Tamanho médio dos pacotes	889	70,143	70	100	196	936,854	68
Bytes/segundo	198877,488	1749,891	2854,553	6618,367	94,147	11594,577	224,84
Protocolos	99,11% TCP	89,19% ARP	52,31% TCP	99,99% TCP	76,32% TCP	97,08% TCP	53,79% ARP
			22,30% TCMP	->33,32% TDS	18,42% UDP		18,03% TCP
			25,13% UDP		5,26% ICMP		3,72% ICMP

CAPEC-87	CAPEC-66	CAPEC-469	CAPEC-482	CAPEC-129
797360	6984	581307	419146	9123
6120	41	3785	6980	78
49,546	20,991	70,918	72,749	37,672
123,520	1,953	53,371	95,946	2,071
130	170	154	60	117
16093,186	332,715	8196,861	5761,512	242,171
100% TCP	63,41% TCP	99,68% TCP	99,01% TCP	51,28% UDP
	29,27% ARP			48,15% TCP

Tabela E.1- Tabela comparativa de tráfego recolhido

Apêndice F

F. Exemplo de um *Dataset*

"No.","Time","Source","Destination","Protocol","Length","Info"

"1","2014-11-06 03:08:42.068884","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has 192.168.56.2? Tell 192.168.56.105"

"2","2014-11-06 03:08:43.055457","c4:00:1b:a4:00:01","c4:00:1b:a4:00:01","LOOP","60","Reply"

"3","2014-11-06 03:08:43.067567","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has 192.168.56.2? Tell 192.168.56.105"

"4","2014-11-06 03:08:44.067625","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has 192.168.56.2? Tell 192.168.56.105"

"5","2014-11-06 03:08:46.069731","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has 192.168.56.2? Tell 192.168.56.105"

"6","2014-11-06 03:08:46.638895","192.168.0.2","192.168.56.105","TCP","74","37642 > 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2364894 TSecr=0 WS=1024"

"7","2014-11-06 03:08:46.639395","192.168.56.105","192.168.0.2","TCP","74","80 > 37642 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=903898 TSecr=2364894 WS=128"

"8","2014-11-06 03:08:46.658874","192.168.0.2","192.168.56.105","TCP","66","37642 > 80 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=2364899 TSecr=903898"

"9","2014-11-06 03:08:46.668837","192.168.0.2","192.168.56.105","HTTP","515","GET /DVWA/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1 "

"10","2014-11-06 03:08:46.669310","192.168.56.105","192.168.0.2","TCP","66","80 > 37642 [ACK] Seq=1 Ack=450 Win=30080 Len=0 TSval=903905 TSecr=2364899"

"11","2014-11-06 03:08:46.670905","192.168.56.105","192.168.0.2","TCP","1514","[TCP segment of a reassembled PDU]"

"12","2014-11-06 03:08:46.671091","192.168.56.105","192.168.0.2","HTTP","337","HTTP/1.1 200 OK (text/html)"

"13","2014-11-06 03:08:46.671330","192.168.56.105","192.168.0.2","TCP","66","80 > 37642 [FIN, ACK] Seq=1720 Ack=450 Win=30080 Len=0 TSval=903905 TSecr=2364899"

"14","2014-11-06 03:08:46.698961","192.168.0.2","192.168.56.105","TCP","66","37642 > 80 [ACK] Seq=450 Ack=1449 Win=32768 Len=0 TSval=2364909 TSecr=903905"

"15","2014-11-06 03:08:46.708902","192.168.0.2","192.168.56.105","TCP","66","37642 > 80 [ACK] Seq=450 Ack=1720 Win=35840 Len=0 TSval=2364911 TSecr=903905"

"16","2014-11-06 03:08:46.718836","192.168.0.2","192.168.56.105","TCP","66","37642 > 80 [FIN, ACK] Seq=450 Ack=1720 Win=35840 Len=0 TSval=2364911 TSecr=903905"

"17","2014-11-06 03:08:46.719378","192.168.56.105","192.168.0.2","TCP","66","80 > 37642 [ACK] Seq=1721 Ack=451 Win=30080 Len=0 TSval=903917 TSecr=2364911"

"18","2014-11-06 03:08:46.728520","192.168.0.2","192.168.56.105","TCP","66","37642 > 80 [ACK] Seq=451 Ack=1721 Win=35840 Len=0 TSval=2364914 TSecr=903905"

"19","2014-11-06 03:08:46.738879","192.168.0.2","192.168.56.105","TCP","74","37643 > 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2364918 TSecr=0 WS=1024"

"20","2014-11-06 03:08:46.739598","192.168.56.105","192.168.0.2","TCP","74","80 > 37643 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=903922 TSecr=2364918 WS=128"

"21","2014-11-06 03:08:46.758999","192.168.0.2","192.168.56.105","TCP","66","37643 > 80 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=2364924 TSecr=903922"

"22","2014-11-06 03:08:46.768814","192.168.0.2","192.168.56.105","HTTP","515","GET /DVWA/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1 "

"23","2014-11-06 03:08:46.769401","192.168.56.105","192.168.0.2","TCP","66","80 > 37643 [ACK] Seq=1 Ack=450 Win=30080 Len=0 TSval=903930 TSecr=2364924"

"24","2014-11-06 03:08:46.770618","192.168.56.105","192.168.0.2","TCP","1514","[TCP segment of a reassembled PDU]"

"25","2014-11-06 03:08:46.770787","192.168.56.105","192.168.0.2","HTTP","337","HTTP/1.1 200 OK (text/html)"

"26","2014-11-06 03:08:46.770998","192.168.56.105","192.168.0.2","TCP","66","80 > 37643 [FIN, ACK] Seq=1720 Ack=450 Win=30080 Len=0 TSval=903930 TSecr=2364924"

"27","2014-11-06 03:08:46.798925","192.168.0.2","192.168.56.105","TCP","66","37643 > 80 [ACK] Seq=450 Ack=1449 Win=32768 Len=0 TSval=2364934 TSecr=903930"

"28","2014-11-06
03:08:46.809055","192.168.0.2","192.168.56.105","TCP","66","37643 > 80 [ACK]
Seq=450 Ack=1720 Win=35840 Len=0 TSval=2364936 TSecr=903930"

"29","2014-11-06
03:08:46.818912","192.168.0.2","192.168.56.105","TCP","66","37643 > 80 [FIN, ACK]
Seq=450 Ack=1720 Win=35840 Len=0 TSval=2364936 TSecr=903930"

"30","2014-11-06 03:08:46.819479","192.168.56.105","192.168.0.2","TCP","66","80 >
37643 [ACK] Seq=1721 Ack=451 Win=30080 Len=0 TSval=903942 TSecr=2364936"

"31","2014-11-06
03:08:46.828887","192.168.0.2","192.168.56.105","TCP","66","37643 > 80 [ACK]
Seq=451 Ack=1721 Win=35840 Len=0 TSval=2364939 TSecr=903930"

"32","2014-11-06
03:08:47.068190","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has
192.168.56.2? Tell 192.168.56.105"

"33","2014-11-06
03:08:48.067432","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has
192.168.56.2? Tell 192.168.56.105"

"34","2014-11-06
03:08:51.070868","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has
192.168.56.2? Tell 192.168.56.105"

"35","2014-11-06
03:08:52.068280","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has
192.168.56.2? Tell 192.168.56.105"

"36","2014-11-06
03:08:53.058738","c4:00:1b:a4:00:01","c4:00:1b:a4:00:01","LOOP","60","Reply"

"37","2014-11-06
03:08:53.067921","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has
192.168.56.2? Tell 192.168.56.105"

"38","2014-11-06
03:08:57.070661","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has
192.168.56.2? Tell 192.168.56.105"

"39","2014-11-06
03:08:58.068142","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has
192.168.56.2? Tell 192.168.56.105"

"40","2014-11-06
03:08:59.067855","CadmusCo_3b:eb:7d","Broadcast","ARP","60","Who has
192.168.56.2? Tell 192.168.56.105"

"41","2014-11-06
03:09:03.059816","c4:00:1b:a4:00:01","c4:00:1b:a4:00:01","LOOP","60","Reply"