View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Repositório da Universidade Nova de Lisboa

Pedro Miguel Bairrão de Seixas Camelo

Licenciado em Engenharia Informática

Botnet Cluster Identification

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Orientadores : Ludwig Krippahl, Professor Auxiliar, Universidade Nova de Lisboa
João Gouveia, Chief Technology Officer, Anubis Networks

Júri:

Presidente: Prof. Doutor Sérgio Marco Duarte

Arguente: Prof. Doutor Rui Alberto Pimenta Rodrigues

Vogal: Prof. Doutor Ludwig Krippahl

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

Botnet Cluster Identification

Copyright © Pedro Miguel Bairrão de Seixas Camelo, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Para ti Géninha e para quem te chamava assim.

Agradecimentos

Esta dissertação não teria sido possível sem o apoio dos meus orientadores Ludwig Krippahl e João Gouveia que me apoiaram no desenvolvimento desta dissertação, à Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa por ser um dos locais que me acompanhou em termos formativos até ao culminar da tese, à AnubisNetworks pelo inesgotável apoio, tanto em termos de formação especifica como pelo espírito aberto para a investigação. Agradeço também ao meu colega João Moura pelo acompanhamento ao longo do percurso da dissertação, aos meus colegas Tiago Martins, Emanuel Alves e Eduardo Duarte por me aturarem e por irem insistindo comigo naqueles dias em que os resultados pareciam não aparecer, aos meus colegas João Gouveia e Valter Santos pelo valioso *insight* relativo ao tema de deteção de *botnets* e práticas comuns de quem as desenvolve e ao meu colega José Ferreira por todo *insight* relativo à deteção de *botnets* através de DNS. Relativamente à escrita do artigo para a Botconf, agradeço aos meus colegas Bruno Rodrigues e Henrique Aparício pelas revisões extensas ao documento. Por fim agradeço aos meus Pais por todo o apoio durante esta fase da minha vida e aos meus amigos pela compreensão.

Resumo

As *botnets* são conjuntos de dispositivos infetados por uma determinada variante de uma família de aplicações maliciosas que responde a um responsável denominado de *botmaster*. Este tipo de redes são usadas para atos ilícitos, onde se inclui extorsão virtual, campanhas de *spam* e roubo de identidade. São usados diversos tipos de mecanismos de evasão que dificultam a deteção e agrupamento de tráfego proveniente de *clusters* associados a *botnets*.

Esta dissertação apresenta uma metodologia denominada de CONDENSER, que forma *clusters* através do uso de um mapa auto-organizável e identifica nomes de domínios gerados por uma semente pseudo-aleatória que é do conhecimento dos *botmasters* responsáveis pelas *botnets*. Adicionalmente foi feito o DNS Crawler, que é um sistema de histórico de DNS, que permite detetar redes em *fast-flux* e *double fast-flux*, assim como relacionar IPs de C&Cs usados em *botnets* através do informação histórica de domínios. Para auxilio e automação da verificação da performance e de seleção de atributos no classificador de DGAs foi desenvolvido o CHEWER que de forma autónoma devolve o melhor conjunto de atributos e parâmetros da SVM para classificação de nomes de domínios associados a DGAs. Tanto o CONDENSER como o DNS Crawler são propostos como soluções escaláveis com o objetivo de melhorar a eficácia de deteção e visibilidade aquando de alterações rápidas na informação nos domínios. Para o classificador de nomes de domínio foi usada uma SVM e é proposto um total de 11 atributos, foi obtido uma Precisão de 77,9% e um valor de *F-Measure* correspondente a 83,2%, tendo sido feita uma seleção de atributos que identificou 3 atributos como tendo mais relevância na classificação. Para o processo de *clustering* foi usado um mapa auto-organizável com um total de 81 atributos.

Foi submetido e aceite um artigo relativo ao trabalho e conclusões desta dissertação, na *Botconf*. A *Botconf* é uma conferência do ramo da investigação, mitigação e descoberta de *botnets* direcionada à indústria, onde é apresentado trabalho e investigação do tema. Esta conferência conta com a presença de empresas de segurança e anti-virus, agências defesa governamental e de investigadores.

Palavras-chave: cibersegurança, botnet, aplicações maliciosas, aprendizagem automática, *clustering*, máquina de vetor de suporte, rede neuronal, seleção de atributos, mapas auto-organizáveis, dns, análise de aplicações maliciosas

Abstract

Botnets are a group of computers infected with a specific sub-set of a malware family and controlled by one individual, called botmaster. This kind of networks are used not only, but also for virtual extorsion, spam campaigns and identity theft. They implement different types of evasion techniques that make it harder for one to group and detect botnet traffic.

This thesis introduces one methodology, called CONDENSER, that outputs clusters through a self-organizing map and that identify domain names generated by an unknown pseudo-random seed that is known by the botnet herder(s). Aditionally DNS Crawler is proposed, this system saves historic DNS data for fast-flux and double fast-flux detection, and is used to identify live C&Cs IPs used by real botnets. A program, called CHEWER, was developed to automate the calculation of the SVM parameters and features that better perform against the available domain names associated with DGAs. CONDENSER and DNS Crawler were developed with scalability in mind so the detection of fast-flux and double fast-flux networks become faster.

We used a SVM for the DGA classififer, selecting a total of 11 attributes and achieving a Precision of 77,9% and a F-Measure of 83,2%. The feature selection method identified the 3 most significant attributes of the total set of attributes. For clustering, a Self-Organizing Map was used on a total of 81 attributes.

The conclusions of this thesis were accepted in Botconf through a submited article. Botconf is known conferênce for research, mitigation and discovery of botnets tailled for the industry, where is presented current work and research. This conference is known for having security and anti-virus companies, law enforcement agencies and researchers.

Keywords: cibersecurity, botnet, malware, machine learning, clustering, support vector machine, neural network, feature selection, self-organizing map, dns, malware analysis

Conteúdo

Lista de Figuras

Lista de Tabelas

Listagens

1

Introdução

Adjacente ao avanço tecnológico, à evolução da Internet e de novas formas de comunicação entre dispositivos, a preocupação com fatores de segurança aumenta. A infeção de dispositivos eletrónicos levantam problemas a nível de privacidade e de segurança de sistemas críticos, factos que tornam a investigação em torno de ameaças que se propagam em rede uma prioridade. As *botnets* são sistemas bastante poderosos em termos de poder de ataque, calcula-se que cerca de 16% a 25% [AMLJS09] do tráfego mundial na Internet seja proveniente de comunicações entre diversos tipos de aplicações maliciosas. Por outro lado observa-se um crescimento [Sma] do número de dispositivos eletrónicos associados a uma pessoa, os *smartphones* e *tablets* são cada vez mais substitutos dos tradicionais telemóveis e computadores, é assim possível dispor de mais poder computacional e consequentemente mais oportunidades de desenvolver aplicações maliciosas que explorem funcionalidades mais vastas no âmbito de cada dispositivo.

As *botnets* são conjuntos de dispositivos, adiante denominados de *clusters*, interligados por rede e operados por um ou mais indevidos para fins ilícitos, também denominados de *botmasters*. A formação de este tipo de redes acontece com o objetivo de fornecer serviços que se aproveitem dos computadores infetados a terceiros. Geralmente cada conjunto de ações, que tiveram origem em um cliente ou que são levadas a cabo com um objetivo especifico é denominado de campanha.

O objectivo desta dissertação é a de apresentar uma forma de classificar nomes de domínios geralmente associados a algoritmos de geração de nomes de domínios (DGA) e de agrupar os dados provenientes de uma fonte de tráfego de internet, de acordo com a sua semelhança. Ambos os objectivos são solucionados com uma abordagem que incide sobre a aprendizagem automática através de um algoritmo de classificação e um de *clustering*.

O sistema desenvolvido não é autónomo, como requisito, mas enquadra-se nos métodos desenvolvidos internamente pela AnubisNetworks. Assim o objectivo é o de auxiliar e facilitar a operação dos analistas e não substituí-lo.

1.1 Motivação

As *botnets* incorrem frequentemente em propósitos ilegais tanto de forma direta como indireta. Estas redes são usadas para roubo de identidade, envio massivo de *spam*, ataques distribuídos de negação de serviço, espionagem governamental, espionagem industrial e uso não autorizado de recursos computacionais, como para *bitcoin mining*[1]. Estatísticas estimam que cerca de 75.2% [Tru13] do tráfego mundial de e-mail é proveniente de *spam* e que a maior parte do tráfego tem origem em *botnets*. Em teoria, este tipo de redes também tem grande potencial em ambientes de guerra e terrorismo, sendo este mais um motivo para travar os autores que dão origem às aplicações maliciosas e consequentemente às *botnets*. Existem diversas autoridades, empresas e organizações que lutam diariamente contra os seus autores e contra a disseminação de aplicações maliciosas.

Atualmente existe mais sensibilidade para os problemas adjacentes à privacidade dos utilizadores nas plataformas on-line e no geral, no entanto ainda não existe uma sensibilização suficiente a fim de tomar precauções para os riscos de infeção. Por outro lado as técnicas de engenharia social[2] são bastante persuasivas e levam a que as suas vitimas instalem aplicações maliciosas nos seus dispositivos ou paguem quantias a fim de reaver dados, que se tornaram inacessíveis após o contágio [Sym14]. O número de escândalos relativos ao uso desmesurado de técnicas questionáveis por parte de agências governamentais tem aumentado, pode observar-se o uso de *botnets* para fins de espionagem a empresas [Bel] recorrendo ao uso de aplicações maliciosas.

1.2 Contexto

A identificação de *botnets* pode ser feita recorrendo a uma análise das aplicações maliciosas numa máquina virtual, com recurso a um ambiente controlado (*sandboxed*), ou através da comunicação que as aplicações efetuam com o C&C ou com outros dispositivos infetados. Existem duas formas genéricas de fazer a deteção, a deteção ativa e passiva. A deteção passiva baseia-se na análise de pacotes para a descoberta de *botnets* e dos seus C&Cs, a deteção ativa baseia-se na injeção de pacotes na rede que podem fazer com que a *botnet* e o C&C sejam revelados. A deteção passiva tem vantagens em relação à ativa já que não alerta o *botmaster* da descoberta iminente da infra-estrutura da *botnet*.

A aprendizagem automática é um ramo da inteligência artificial que estuda a aprendizagem de conceitos, comportamentos ou valores por parte de dispositivos eletrónicos.

[1] Moeda virtual criada por Satoshi Nakamoto, o termo *mining* refere-se à descoberta de provas matemáticas com recurso a dispositivos eletrónicos que origina valor para o seu utilizador.

[2] Este termo no contexto de segurança refere-se à manipulação psicológica de um indivíduo para efetuar ações ou divulgar informações privadas.

No ramo de segurança eletrónica e de informação, destaca-se a identificação de tráfego malicioso, deteção de *spam* e deteção de aplicações maliciosas.

Esta dissertação é feita para a obtenção do grau de Mestre em Engenharia Informática, pela Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (FCT–UNL) em colaboração com a empresa prepotente, a AnubisNetworks.

1.2.1 AnubisNetworks

A AnubisNetworks é uma marca da NSEC, Sistemas Informáticos, SA. Esta empresa criada em 2006, atua sobre o ramo de Internet fornecendo soluções anti-*spam*, controlo parental, *security intelligence* e análise de aplicações maliciosas. Os seus clientes passam por operadores de comunicações, bancos e agências de defesa e informação. Em conjunto com a contribuição para a academia e resposta ao crescente número de *botnets*, nasceu uma colaboração entre a Faculdade de Ciências e de Tecnologia da Universidade Nova de Lisboa e a AnubisNetworks com vista a melhorar os produtos disponibilizados. Parte do trabalho efetuado nesta dissertação irá assentar sobre produtos já disponibilizados pela AnubisNetworks, nomeadamente o *Stream Platform Service*, o *MailSpike* e o *Maltracker*.

1.2.1.1 *Stream Platform Service*

O *Stream Platform Service* (adiante designada de *Stream*) é uma plataforma de distribuição de mensagens produzidas por serviços e consumidas por assinantes. Os eventos produzidos tem origem em diversas fontes. A *Stream* dispõe de funcionalidades de agregação, filtragem e cálculo sobre os dados produzidos. Existem diversos canais (adiante designados de *feeds*) de distribuição de conteúdo, sendo que esta dissertação assenta sobre três tipos de canais: tráfego rede contendo tráfego indiscriminado; tráfego de rede contendo tráfego malicioso; tráfego rede correspondente a acesso a NXDOMAINS, discutido no sub-capítulo 2.1.5.2.

1.2.1.2 *Maltracker*

O *Maltracker* (`https://maltracker.net`) é um sistema de análise de aplicações maliciosas que permite submissão de ficheiros para análise forense de ameaças que se disseminam a partir de ficheiros contendo aplicações maliciosas. Esta plataforma permite fazer um acompanhamento das infeções, dispositivos infetados, análise rede e deteção de C&C em tempo real. Esta plataforma tem a vantagem de assentar sobre uma deteção ativa, sendo portanto bastante precisa e com informação de bastante veracidade. Também dispõe de um serviço de análise de sítios da Internet que permite obter informações da comunicação e interação de sítios com o objetivo de comprometer os seus utilizadores ou de venda de objetos ilícitos, ou de forma ilícita.

1.2.1.3 *MailSpike*

O *MailSpike* (http://mailspike.org/) é um serviço de reputação IP que identifica *spamers* conhecidos, é um serviço gratuito e atualizado em tempo real, com informações de IPs conhecidos por vagas de *spam* recentes. Este serviço vai ser acedido através de um modulo adicional da *Stream*, podendo não haver uma reputação associada a todos os IPs observados durante a análise.

1.3 Estrutura do Documento

Este documento está dividido em cinco capítulos, a introdução, onde se introduziu a motivação, o contexto e onde se fez uma introdução ao tema das *botnets*. O trabalho relacionado, onde é aprofundado as características das botnets, se faz o levantamento da investigação existente no âmbito do tema em estudo e as soluções atuais para o problema descrito. As contribuições, onde se descreve o trabalho efetuado para a obtenção de resultados e soluções para o problema proposto. Os resultados, onde são descritos os resultados obtidos para os métodos usados nesta dissertação. As conclusões, onde se discute o resultado do trabalho efetuado, limitações e trabalho futuro. No apêndice encontram-se tabelas e outra informação adicional relativa a valores obtidos no capítulo dos resultados e atributos usados para o treino dos algoritmos de aprendizagem automática.

2

Estado da Arte

A investigação sobre o tema das *botnets* no ambiente académico é recente [SSPS13], pode-se observar um crescimento considerável na investigação relacionada com a deteção e estudo de *botnets* desde o ano de 2005. Este capítulo pretende fazer uma análise da literatura existente sobre as formas de deteção de *clusters* associados a *botnets* e encontra-se dividido em seis secções.

Na primeira secção é feita uma análise às características e funcionamento das *botnets*, na segunda é feito um levantamento das formas de deteção existentes, na terceira é feito um levantamento de algoritmos de aprendizagem automática que foram úteis para esta dissertação e na quinta é descrito os algoritmos de *clustering* considerados.

2.1 O que é uma *Botnet* ?

Ao longo do levantamento do trabalho relacionado foi notado que a definição de *botnet* e de uma aplicação maliciosa pode ter várias interpretações, não havendo uma definição que cubra todo o escopo de operação de uma *botnet* e de uma aplicação maliciosa do ponto de vista do autor. São assim descritas as assunções sobre os dois termos e o contexto em que é referido ao longo deste documento. Assim, tem-se em conta as seguintes definições informais.

Definição Informal 1. *Assume-se que uma aplicação é maliciosa a partir do principio que está envolvida em atividade ilegal ou potencialmente prejudicial para o(s) proprietário(s) do dispositivo infetado e que foi instalada no dispositivo alvo sem conhecimento, permissão e consciência das suas*

acções[1] *[LM]. Resguarda-se a existência de sistemas com objetivos legítimos que possam ter comportamentos semelhantes aos de uma aplicação maliciosa, ou de uma botnet, também denominados de falso-positivos.*

Definição Informal 2. *Assume-se que uma botnet, é o conjunto do(s) centro(s) de controlo, eventuais dispositivos intermédios para ocultação do centro de controlo, e dispositivos finais (ex: computadores, telemóveis). Considera-se que uma botnet encontra-se associada a uma ou mais pessoas que requisitam [SGCCGSKKV09] ou iniciaram um determinado ataque criminoso direcionado com um certo objetivo*[2]*. A determinação da botnet também se encontra associada a uma determinada versão de uma família de aplicações maliciosas (Zeus e Rimecud por exemplo).*

Conforme descrito na Definição Informal 2, a definição de *botnet* pressupõem o acesso a de computadores infetados a terceiros como um serviço, este tipo de serviço também é chamado de *Botnet as a Service* [SGCCGSKKV09]. Dado que a deteção do fornecimento tal serviço é complicado de forma passiva e sem interação ou exploração de vulnerabilidades do C&C, formou-se a Definição Informal 2.1, esta definição encontra-se contida na Definição Informal 2 e relaxa o escopo do significado de uma *botnet* ao longo deste documento.

Definição Informal 2.1. *Assume-se que uma botnet, é o conjunto do(s) centro(s) de controlo, eventuais dispositivos intermédios para ocultação do centro de controlo, e dispositivos finais (ex: computadores, telemóveis). A determinação da botnet também se encontra associada a uma determinada versão de uma família de aplicações maliciosas (Zeus e Rimecud por exemplo).*

Este capítulo discute a forma como as *botnets* operam e que características apresentam. Estes tipos de *cluster* apresentam uma arquitetura cliente e servidor, que pode estar disposta em diferentes topologias, tópico discutido na secção 2.1.2. Apresentam também mecanismos de evasão à deteção e de prevenção a ataques com vista a interromper o seu funcionamento, este tópico é discutido no secção 2.1.5. Para poder ser criada, necessita de um modo de disseminação e contágio, na secção seguinte apresenta-se um modelo comum presente nas *botnets* existentes. Salvaguarda-se a existência de outro tipos de disseminação e modelos de funcionamento não descritos neste documento, já que os autores procuram resiliência e métodos de disseminação cada vez mais avançados para a sua prática sem levantar suspeitas.

2.1.1 Modo de Operação e Ciclo de Vida

Para enquadramento completo do funcionamento de uma *botnet* no contexto actual relembre-se da Definição Informal 2, em que é referido a existência de *botnets* como um serviço. Destacam-se dois modos de operação: no primeiro uma *botnet* é criada com o objetivo específico de um singular ou de um coletivo de criminosos que pretendem efetuar algum

[1] RFC 6561: *Recommendations for the Remediation of Bots in ISP Networks* - `http://tools.ietf.org/html/rfc6561`

[2] No ramo de segurança informática este tipo de ataques é denominado de campanha.

tipo de operação ilegal, o segundo modo de operação baseia-se na criação de uma infra-estrutura que pode ser vendida a clientes finais que pagam pelo acesso aos computadores infetados. Por este motivo a Definição Informal 2.1 faz mais sentido na medida que a deteção de uma *botnet* segundo a Definição Informal 2 é um processo difícil, por envolver uma interação direta com o C&C e entidades legais que investigam a atividade criminosa adjacente ao grupo (ou individual) que opera toda a infra-estrutura. Por outro lado foi também observado [SGCCGSKKV09] que as aplicações maliciosas apresentam esporadicamente identificadores da campanha correspondente, reforçando mais uma vez a existência da Definição Informal 2.

As *botnets* apresentam um ciclo de vida próprio geralmente composto por cinco fases [SSPS13]. Primeiro, o autor dissemina uma versão inicial da aplicação maliciosa, que permite iniciar o ciclo de infeção e formar gradualmente a *botnet*. A sua propagação é feita através de sites comprometidos, como plataformas de gestão de conteúdos também conhecidas por *Content Management Systems* (CMS) e campanhas de *spam* com recurso a computadores infetados. Geralmente a infeção é feita com técnicas de engenharia social, com fim de iludir o utilizador para a instalação de tais programas. Na segunda fase é descarregada uma aplicação com capacidades mais avançadas do que a aplicação inicial. Na terceira fase é feita a comunicação com o centro de controlo onde existem mecanismos de recuperação que têm o objetivo de garantir que esta comunicação é bem sucedida. Na quarta fase é executada a atividade maliciosa, em que se pode incluir roubo de identidade, campanhas de *spam*, ataques distribuídos de negação serviço, entre outros. Por se tratar de um modelo que permite a disseminação de aplicações a outros computadores e pela consumação crime, esta trata-se da fase mais critica e vantajosa para os *botmasters*. A quinta e última fase corresponde à atualização e manutenção das aplicações maliciosas, permitindo que versões melhoradas da aplicação sejam instaladas nos dispositivos comprometidos.

2.1.2 Arquiteturas Existentes

As *botnets* podem dispor-se em quatro tipos de arquiteturas: centralizada, distribuída, híbrida e aleatória. A sua composição e características variam de arquitetura para arquitetura, sendo que apresentam geralmente mecanismos de falha que podem fazer com que variem entre arquiteturas, sendo que a variação de arquiteturas também pode ser considerado um método de evasão, como irá ser discutido na secção 2.1.5.

2.1.2.1 Arquitetura Centralizada

A arquitetura Centralizada representada na Figura 2.1 é composta por um C&C central que se encarrega de comunicar e coordenar a *botnet*, este tipo de arquitetura também é chamada de arquitetura em estrela. Para este género de arquiteturas existe um subgénero que é identificado pela rotação do C&C principal, representado na Figura 2.2.

Este sub-género de arquitetura também é apelidada por *Locomotive Architecture* na literatura existente. As *botnets* que seguem este tipo de arquitetura apresentam um modo de funcionamento semelhante, ou mesmo igual a protocolos usados na Internet como é o caso do HTTP.

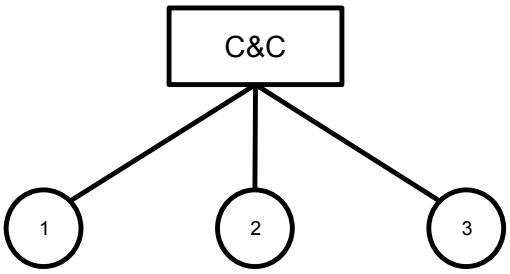


Figura 2.1: Arquitetura Centralizada

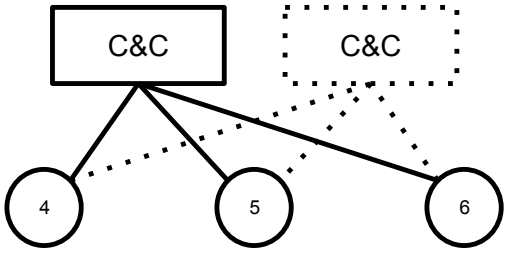


Figura 2.2: Arquitetura Centralizada com Rotação do C&C

2.1.2.2 Arquitetura Distribuida

A Arquitetura Distribuída é composta por vários C&Cs e por dispositivos comprometidos, que comunicam entre si. Geralmente este tipo de arquiteturas recorre a protocolos geralmente associados ao BitTorrent[3], como é o caso da DHT. As *Distributed Hash Table* é um algoritmo distribuído para descoberta de dispositivos na rede, não necessitando de um servidor central para o seu funcionamento.

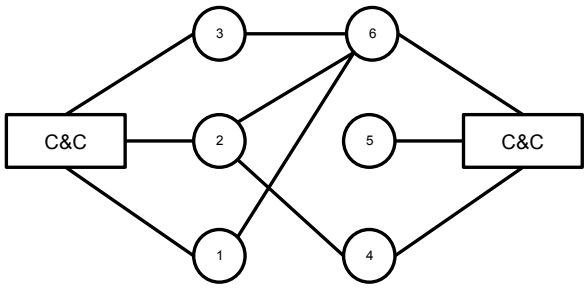


Figura 2.3: arquitetura Distribuida

2.1.2.3 Arquitetura Híbrida

A arquitetura Híbrida é composta por uma mistura da arquitetura centralizada e da arquitetura distribuída. Este género de arquitetura dispõem de diversos C&Cs – à semelhança da arquitetura distribuída – que são contactados por dispositivos comprometidos que atuam como *proxy* para outros dispositivos infetados pertencente à mesma botnet. Neste tipo de arquitetura também são usados protocolos e infra-estruturas que anonimizam a operação da *botnet*, como o *onion routing*, presente no software de *routing* de

[3]Protocolo de troca de ficheiros de grande dimensão, criado em 2001 por Bram Cohen, usado inicialmente para *download* de distribuições *Linux* - http://en.wikipedia.org/wiki/BitTorrent

tráfego de forma anónima, Tor. Este tipo de encaminhamento de tráfego envia os pacotes rede por caminhos aleatórios, dificultando o acompanhamento da origem e destino do tráfego. Por estas razões este tipo de arquiteturas é descrita [SZ10] como a mais resiliente das descritas neste capítulo

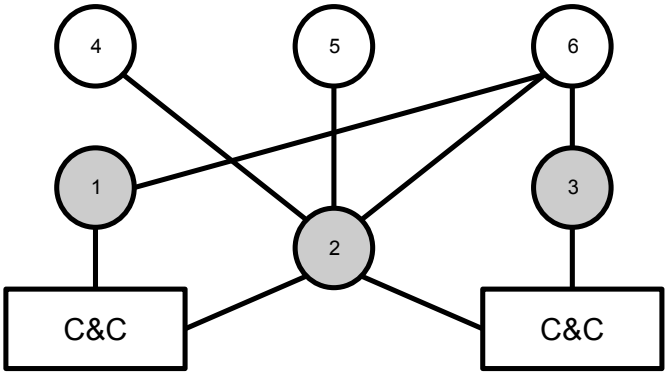


Figura 2.4: Arquitetura Híbrida com representação a cinzento dos dispositivos a funcionar como *proxy*

2.1.2.4 Arquitetura Aleatória

Na Arquitetura Aleatória cada instância da aplicação maliciosa pode-se comportar como um C&C ou como uma instância da versão simples da aplicação maliciosa. A literatura [Hon12] também refere um sub-género para este tipo de arquitetura, como é o caso da arquitetura não estruturada. Para este tipo de arquitetura o *botmaster* efetua uma pesquisa na rede pelos C&Cs disponíveis, tomando de seguida o controlo da *botnet*. Por não ter um mecanismo central de gestão, a monitorização deste tipo de arquitetura torna-se bastante complicada, podendo mesmo existir computadores infetados que nunca cheguem a interagir com a restante *botnet*.

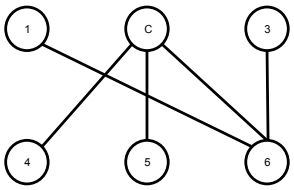


Figura 2.5: arquitetura Aleatória

2.1.3 Sistema de Nomes de Domínios

No contexto das *botnets*, o sistema de nomes de domínios (adiante designado de DNS) é bastante usado tanto para acesso à infra-estrutura da *botnet* como para evasão à deteção (discutido no capítulo 2.1.5). Este sistema criado [Moc83a; Moc83b] em 1983 por Paul Mockapetris é um serviço de resolução de endereços para um nome de domínio e funciona de forma hierárquica, sendo que a base de implementação atual baseia-se nas

convenções de 1987 [Moc87a; Moc87b]. Este protocolo foi criado para que um utilizador possa aceder a serviços na Internet de forma mais prática, sem necessidade de memorizar endereços IP[4] para aceder a serviços na Internet. Para demonstrar o seu funcionamento tome-se o exemplo do domínio `www.example.com`. Neste domínio podem ser observados três níveis hierárquicos, o nível de topo (também referido como *Top Level Domain* - TLD) `com`, o segundo nível `example` e o terceiro nível `www`. Foi efetuada uma consulta DNS, através do comando `dig`, disponível na generalidade das distribuições Linux disponíveis. O resultado da execução do comando pode ser observado na Figura 2.6.

```
$ dig -t ANY www.example.com @8.8.8.8
; <<>> DiG 9.8.3-P1 <<>> -t ANY www.example.com @8.8.8.8
;; global options:  +cmd
;; Got answer:
;; ->>HEADER<<- opcode:  QUERY, status:  NOERROR, id:  25936
;; flags:  qr rd ra; QUERY: 1, ANSWER: 18, AUTHORITY: 0,
ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.com.     IN     ANY

;; ANSWER SECTION:
www.example.com.     21570     IN     A      93.184.216.119
www.example.com.     21536     IN     TXT     "v=spf1 -all"
```

Figura 2.6: Exemplo do comando `dig` usado para obtenção das informações de registos existentes no nome de domínio `www.example.com`.

Para cada domínio podem existir registos que determinam a localização de serviços associados a cada domínio. Estes registos também são denominados de *records* e tem um valor e período de validade associados. Um exemplo de serviço que necessita de um registo especifico para o seu funcionamento com domínios é o caso dos *nameservers*. Os *nameservers* (servidor de consultas de nomes de domínio) são um protocolo de parqueamento de domínios, estes servidores consistem em servidores de alta disponibilidade que fornecem serviços de consulta aos domínios registados na sua base de dados. Na Figura 2.7 pode ser observada uma consulta aos registos correspondentes ao endereço dos *nameservers* do nome de domínio `example.com`. Estes registos são indicados pelo acrónimo da palavra *nameserver*, `NS`. Na Figura 2.7 pode-se observar que o servidor de nomes associado ao domínio `example.com` tem o valor `a.iana-servers.net.` e `b.iana-servers.net.`, com um período de validade de `16499` segundos para cada registo. Para os registos de tipo `NS`, `A`, `SOA` e `SPF` existem valores que podem indicar a existência de uma *botnet* por apresentarem um período de validade curto e por apresentarem valores que geralmente não se encontram em domínios para operações legitimas.

[4]O *Internet Protocol* é um protocolo de identificação de computadores na rede criado por Vint Cerf e Bob Kahn em 1974

```
$ dig -t NS example.com @8.8.8.8
...
;; ANSWER SECTION:
example.com.     16499     IN     NS     a.iana-servers.net.
example.com.     16499     IN     NS     b.iana-servers.net.
```

Figura 2.7: Exemplo do comando `dig` para obtenção dos registos `NS` do domínio `example.com`.

Na secção 2.1.5 são discutidos os mecanismos de evasão à deteção que podem ser associados a valores presentes em informação de DNS, no capítulo 2 são discutidas formas de deteção existentes na literatura que podem indicar a existência e a infra-estrutura de uma *botnet* com recurso a análises de infornação de DNS.

Sinkholing

A técnica de *sinkholing* consiste no registo de nomes de domínio associados a atividade de *botnets*, reencaminhando o tráfego para um destino diferente ao do C&C. Esta técnica é usada para fazer monitorização de *botnets* com o objetivo de perceber a sua topologia e modo de operação. Tendo esta informação é possível determinar a amplitude de infeção, taxas de crescimento, objetivos e os responsáveis pela operação da *botnet*. Esta técnica pressupõe que os responsáveis do mecanismo de *sinkhole* implementado não manipulam as máquinas infetadas que comunicam com o *sinkhole*.

2.1.4 Sistemas de Deteção e Prevenção de Intrusões

A deteção de intrusões é [SM07] o processo de monitorização de eventos que ocorrem num sistema computacional ou em rede, e na respetiva análise para descoberta de possíveis *incidentes*. Por sua vez, um sistema de de deteção de intrusões, adiante designado de IDS (*Intrusion Detection System*), é uma aplicação que automatiza o processo de deteção de intrusões. A nível de referência futura também se define os sistemas de prevenção de intrusões, adiante designados de IPS (*Intrusion Prevention System*), que consiste numa aplicação que elimina ativamente as ameaças detetadas, neste último caso, com recurso às técnicas de sistemas como o IDS. Este tipo de sistemas usam deteção baseada em: assinaturas, em que o fragmento dos dados são comparados com padrões já conhecidos; anomalias, em que os dados fogem a um conjunto de dados considerado normal; e por estado, em que a sequência dos dados é comparada com protocolos já conhecidos. Alguns dos métodos descritos no trabalho relacionado, discutidos na Secção 2.2, são comparados com IDS e IPS a nível de performance e eficácia de deteção.

2.1.5 Mecanismos de Evasão à Deteção

Para garantir o funcionamento e permanência da *botnet*, os *botmasters* implementam formas de ocultar o seu funcionamento. Hoje em dia os mecanismos de evasão à deteção são cada vez mais avançados e originais, sendo que o nível de sofisticação encontrado varia

de acordo com o propósito da *botnet*. Famílias de aplicações maliciosas como o *Cryptolocker*[5] implementam diversos tipos de mecanismos de evasão e ocultação do seu funcionamento e infraestrutura. Nas próximas secções pretende-se efetuar um levantamento dos mecanismos observados na literatura existente assim como algumas referências mais pontuais.

2.1.5.1 Comunicação Cifrada

Na implementação de aplicações maliciosas é comum o uso de mecanismos de cifra dos dados, quer por algoritmos de criptografia mais usuais quer por codificação da informação da comunicação com o C&C (e.g *base64*). O uso de codificações que não usuais também torna o processo de identificação mais complicado. O uso de comunicação cifrada entre o C&C e os dispositivos infetados leva a que o seu conteúdo não possa ser tomado em consideração aquando de extração de atributos relevantes para a deteção de tráfego anómalo.

Codificação em Base64 (original): `/Q9NDA1OTImc2l(...)kPTAmcmQ9MA==37A`
Codificação em UTF-8: `(...)clk=X.2bid=89(...)73aid=XXXXXsid=Xrd=X`

Figura 2.8: Demonstração de um exemplo real da comunicação da Família de aplicações maliciosas *TDSS/Clicker* que codifica a informação presente no URI em Base64.

Na Figura 2.8 pode-se observar um exemplo de aplicação de uma cifra na comunicação originada por uma aplicação maliciosa. Para garantir a privacidade do utilizador em questão foi omitida alguma informação do exemplo apresentado. No entanto a extração de meta-dados do conteúdo, como o comprimento do conteúdo do pacote e outros atributos semelhantes mostram ter resultados [KR10b; KR10a; AB11] o que permite usar atributos relevantes para a deteção, como será discutido na Secção 2.2. Por vezes também é possível extrair informações relativas aos certificados usados pelas aplicações maliciosas para cifra dos dados, através de engenharia reversa, este facto permite detetar e revelar a própria estrutura e topologia da *botnet*.

2.1.5.2 Algoritmos de Geração de Nomes de Domínio

Os Algoritmos de Geração de Nomes de domínio, adiante designados de DGAs (*Domain Generation Algorithms*) são algoritmos pseudo-aleatórios alimentados por uma semente geralmente desconhecida ou imprevisível, por vezes obtidas através dos tópicos populares de serviços como o *Twitter* e os primeiros resultados de pesquisas do *Google* [SMCZ13].

Os DGAs são usados para gerar nomes de domínio com caracteres aleatórios ou baseados em dicionários de palavras reais, com o objetivo de esconder a localização IP do C&C, e desta forma dificultar ações contra a infra-estrutura da *botnet*. Os DGAs são um

[5]Família de aplicações maliciosas que encripta os ficheiros relevantes existentes num computador infetado. Para reverter a cifra dos ficheiros é necessário pagar um resgate ou obter as chaves de encriptação necessárias.

mecanismo de evasão importante no contexto das *botnets* já que geralmente apresentam um compromisso para o seu autor: grande parte dos nomes de domínio gerados não resolvem para nenhum IP e consequentemente geram respostas de domínio não existente, adiante designadas de `NXDOMAIN` (*Non-Existent Domain*), por parte dos servidores DNS. A existência de pedidos de resolução consequente a domínios que geram `NXDOMAINS` é um comportamento que não se verifica geralmente em dispositivos sem qualquer tipo de infeção, conforme discutido em [AP12]. No entanto alerta-se o leitor que existem aplicações legitimas que podem gerar `NXDOMAINS`. Ao longo da dissertação foi observado que o navegador de Internet *Google Chrome* tem uma DGA na sua implementação, pelo que foi possível observar a geração de falsos positivos em rede. Esta DGA existe[6] para verificar se o provedor de serviços de Internet do utilizador faz uma substituição das respostas `NXDOMAIN` por respostas de domínios existentes para inserção de conteúdos publicitários, ou se alguma extensão ou plugin interceta o pedido por outros motivos.

```
cdjtjcj.info          frbyls.net            fkqqrqxtkxfi.info
bqtcdqacrgk.info      evphbxaojka.info      dvrivq.info
goquhifefgf.info      cxwzlutzra.info       dgfolgtqh.net
iqefpvgbuc.info       cwegttl.info          dcnbvvbyygg.info
hdpkptgcep.info       cifwpcbdu.info        boetulqonqrf.net
gghvpoc.info          bjgkuuf.info          blyujitcx.com
cdbrzxma.info         holakaddbqt.info      ayuemiqi.org
brprvvtbpil.info      gwnxtcxwzdb.net       aabobefad.net
afsoletbtg.net        gbfcco.net            kgdngbi.info
```

Figura 2.9: Exemplo de uma lista de domínios originados por uma DGA.

Na Figura 2.9 demonstra-se uma DGA real, todos os domínios representados originam respostas `NXDOMAIN`, o que leva o autor a crer que estes domínios podem ter sido originados por uma aplicação maliciosa. Chama-se a atenção para a ocorrência aleatória de caracteres nos domínios, fora os TLDs apresentados. É possivel observar particularidades a nível semântico, sendo possível aferir padrões através da observação na ocorrência de caracteres, frequência da ocorrência de consoantes, vogais observadas e análise com recurso a n-gramas, conforme é discutido na Secção 2.2.

2.1.5.3 *Fast-Flux* e *Double Fast-Flux*

As técnicas de *Fast-Flux* e *Double Fast-Flux* consistem na alteração constante de valores presentes em registos DNS com recurso a períodos de validade reduzidos. A técnica de *Fast-Flux* incide exclusivamente em registos associados diretamente ao domínio (e.g `A` e

[6] Apesar de haver inúmeras referências em fóruns da existência desta DGA e do autor ter comprovado a sua existência, não existe nenhum comunicado oficial da *Google* que confirme este facto. No entanto existem referências à problemática do `NXDOMAIN` *hijacking* por parte da empresa no seguinte endereço: *How does Google Public DNS handle non-existent domains?* - https://developers.google.com/speed/public-dns/faq#nxdomains

AAAA), enquanto que a técnica de *Double Fast-Flux* apenas reside na alteração dos registos no endereço do(s) *nameserver* associado ao domínio. Existem casos de uso que podem ser confundidos com *Fast-Flux* como para o caso de redes de distribuição de conteúdo, adiante designadas de CDN (*Content Delivery Network*). Na Figura 2.10 pode-se observar uma alteração do IP no domínio da *Heroku*, uma empresa que fornece serviços de distribuição de aplicações *cloud* na Internet.

```
$ dig -t A herokuapp.com @8.8.8.8
(...)
;; ANSWER SECTION:
herokuapp.com.      59      IN      A      50.17.221.5
(...)
$ dig -t A herokuapp.com @8.8.8.8
(...)
;; ANSWER SECTION:
herokuapp.com.      51      IN      A      107.20.186.77
```

Figura 2.10: Demonstração de um comportamento semelhante a técnicas de *Fast-Flux*, resultado obtido através de uma consulta DNS ao nome de domínio `herokuapp.com` com recurso ao programa `dig`.

Na Figura 2.10 também é possível observar um comportamento semelhante a uma *fast-flux* típica, que poderia ser confundida com uma tática de evasão implementada por uma *botnet*. O registo do tipo `A` presente no domínio `herokuapp.com` apresenta um registo de validade (também denominado de TTL - *Time to Live*) de 59 segundos, onde houve uma alteração do registo IP num curto espaço de tempo. Este comportamento também é verificado em *botnets*, que recorrem a um grande número de IPs para a sua implementação. Este método é geralmente usado em *botnets* com arquiteturas híbridas, em que os *botmasters* usam os IPs de dispositivos infetados para funcionar como *proxy*, auxiliando e escondendo a sua atividade.

2.1.5.4 Outros

Para iludir os sistemas de deteção, são usado mecanismos pouco convencionais que podem ser variações dos métodos anteriormente descritos ou técnicas novas de forma a circunscrever mecanismos de deteção como IPS e IDS. Tome-se o exemplo da Família de aplicações maliciosas *Necurs* que recorre à resolução de um nome de domínio e troca o valor dos bits do resultado da consulta para obter o endereço final [Nora; Norb]. Esta técnica permite iludir a deteção do C&C por métodos diretos, porque neste caso especifico o domínio devolve uma página aparentemente legitima, e mais do que isso, pode usar mesmo endereços de serviços legítimos para obter o endereço do C&C.

Também foi observado [XBLXT11] um tipo de aplicação maliciosa que aplica uma técnica denominada de *URL-Flux* para obter informações das operações que deve efetuar. Neste caso a plataforma usada para C&C são sites legítimos de *Microblogging*, em que os *botmasters* implementam um algoritmo de geração de contas de acesso a essas

páginas (semelhante ao funcionamento das DGAs anteriormente descritas) e registam previamente um utilizador que irá ser gerado no futuro. Mais tarde, com recurso a um *post* ou comentário ordenam a execução de ordens por intermédio de imagens cifradas, ou de comentários com significado relevante para a execução da aplicação maliciosa.

No trabalho [HG13] desenvolvido por Hanspach, et al. foi feita e implementada uma metodologia de troca de mensagens com recurso a ultra-sons, através das colunas dos dispositivos infetados. Este método torna a deteção da comunicação difícil dado que este é um método com características muito especificas e novo no âmbito de técnicas usadas, já que explora o meio físico como troca de mensagens, podendo mesmo infetar dispositivos que não tenham qualquer tipo de ligação à Internet. Tendo em conta mesmo método também se refere o trabalho de [Des14], este artigo foca diversos problemas relativos à dificuldade de deteção de mensagens sobre esta forma de troca de mensagens. Chama-se a atenção de que os método descritos neste parágrafo saem fora do escopo da dissertação por se basear num meio diferente de transmissão ao abordado, pretende-se realçar e incentivar a investigação na área com a sua referência.

2.2 Deteção de *Botnets*

A deteção de *botnets* consiste - confirme definido na Definição Informal 2.1 - na análise e deteção do conjunto do(s) centro(s) de controlo usados na *botnet*, eventuais dispositivos intermédios para ocultação do centro de controlo, e dispositivos finais. Para se poder atribuir corretamente a responsabilidade, ou as aplicações maliciosas responsáveis por esse *cluster*, é necessário a determinar família de aplicações maliciosas que está na origem do mesmo.

Na primeira secção é feita uma análise ao às palavras do nome de domínio (ou ao seu conjunto de caracteres) para deteção de nomes gerados algoritmos de gerações de nomes de domínio, a segunda recai sobre uma análise singular ao conteúdo de cada pacote do tráfego rede, a terceira recai sobre uma análise sobre fluxos de rede, a quarta recai sobre uma análise à informação de registo dos domínios em servidores DNS, a quinta e sexta discutem o uso de Maquinas de Vetor de Suporte e de Redes Neuronais Auto-associativas no contexto abordado nos tópicos anteriores.

2.2.1 Deteção por Caracteres do Domínio

Este tipo de deteção ocorre sobre a análise do nome de domínio (ou ao conjunto de caracteres) para determinar se o mesmo foi gerado por algoritmos pseudo-aleatórios, também denominados de DGAs, normalmente usados para evasão à deteção, como discutido na Sub-Secção 2.1.5.

No estudo feito por [DK13] foi feita uma análise aos nomes de domínio, no seu estudo separou cada nome de domínio pelos seus sub-componentes (TLD,2LD,3LD...[7]) e aplicou

[7]Estas siglas referem-se às várias componentes de um nome de domínio. Por exemplo, no nome de

um algoritmo baseado numa análise por bi-gramas. Este algoritmo originou um atributo de entrada para uma Máquina de Vetor de Suporte. Foi verificado que o uso de uma Máquina de Vetor de Suporte teve melhores resultados do que o uso de *Näive Bayes* ou de *C5.0* para a classificação.

[SMCZ13] realçou diversos problemas importantes no contexto da deteção de domínios maliciosos, tendo sido criado um sistema de deteção que conseguiu identificar corretamente domínios gerados pela aplicação maliciosa *Conficker.B*. Foi considerado que a semelhança e análise linguística de um nome de domínio não é suficiente para determinar se um domínio é malicioso ou não, por esta razão também foram considerados outros atributos ao nível IP e DNS como medida de semelhança.

Frosch, et al. [FMHHB] usou um classificador do tipo *K-Nearest Neighbour*. Para treino benigno usou o top 20.000 da empresa *Alexa*, tendo identificado no seu estudo 40.000 domínios como pertencentes a CDNs. Para treino maligno usou domínios (de três Famílias de aplicações maliciosas diferentes) que se encontravam listados no sitio `http://abuse.ch`.

2.2.2 Deteção por Pacote

Este tipo de deteção ocorre sobre uma análise a cada pacote trocado em rede, a análise pode ocorrer apenas sobre os cabeçalhos do pacote, ou sobre o conteúdo completo do pacote. Cada abordagem tem as suas vantagens, sendo que a análise do cabeçalho dos pacotes é mais rápida quando comparada com o a análise completa de cada pacote [ZTSLSGG13]. Por sua vez a análise completa de pacotes mostra-se mais eficiente quando comparada com a análise dos cabeçalhos.

No estudo feito por [AJS06] é apresentado o uso de redes neuronais para a deteção em intrusão de sistemas informáticos. Foram analisadas três tipos de redes neuronais: *Adaptive Resonance Theory* (ART-1, ART-2) e *Self Organizing Map* (SOM). Foram destacados dois problemas durante a implementação da sua solução, o primeiro estava relacionado com a performance dos resultados ao usar uma representação explicita do tempo (etiqueta do instante de receção - *timestamp*), a solução passou por usar uma representação implícita (alimentar o algoritmo por ordem de chegada, sem representação do tempo). A segunda estava relacionada com o elevado tempo de treino das redes neuronais, por outro lado o tempo de classificação de novas instâncias do tráfego após o treino mostrou ser satisfatório. O melhor resultado obtido foi com o uso de ART-1, com um valor de precisão de 71.17% e com 1.99% de falsos positivos.

Ruehrup, et al. [RU13] usa grafos de ligação a destinos comuns para estabelecer padrões entre as ligação de dispositivos infetados e os C&Cs. Apesar de não usar algoritmos de aprendizagem automática, o uso de grafos de ligação a destinos comuns levanta uma perspetiva interessante que pode ser considerada na criação de atributos adicionais para o tema em estudo.

domínio `foo.bar.com`, o TLD (*Top Level Domain*) corresponde ao conjunto de caracteres ´com¡ o 2LD (2^{nd} *Level Domain*) corresponde ao conjunto de caracteres `bar`, assim sucessivamente.

2.2.3 Deteção por Fluxo de Rede

Um fluxo de rede é um conjunto de atributos, sendo constituído por um conjunto de pacotes compostos pelo mesmo pelo endereço IP de origem e destino, porto de origem e destino, e pelo protocolo ao nível da camada de transporte. Esta análise é mais lenta quando comparada com as anteriores porque pode necessitar de um volume considerável de tráfego para obedecer a um padrão anómalo.

O estudo feito por [BBR12] recai sobre a análise de fluxos de rede através de classificadores *Random Forest*. Para a sua solução recorreram a dois conjuntos de dados obtidos através de uma captura de 18 dias, o primeiro conjunto foi proveniente de uma universidade na Europa, o segundo de um fornecedor de serviços de primeira ordem (*Tier-1*). Foram usadas três plataformas adicionais para apoio à classificação: o serviço FIRE, que é um sistema de reputação de fornecedores de serviços de Internet e organizações, o serviço EXPOSURE, que usa uma análise passiva aos DNS de forma a detetar domínios maliciosos, e o serviço *Google Safe Browsing*, que é um sistema de classificação de sítios de Internet e de sistemas autónomos, da Google. Para o primeiro caso a captura foi completa, para o segundo a captura foi feita com um rácio de 1:10,000. Os resultados obtidos foram de 65% de precisão para 1% de falsos positivos. O facto de apenas ter sido capturado parte do tráfego pode ser interessante para fins de avaliação da implementação desta dissertação, no entanto apresenta a desvantagem de poder perder todos os pacotes referentes ao tráfego de uma *botnet*.

Amoli et. Al [AH13], apresenta um modelo para deteção de ataques em redes de alta velocidade, consideram apenas o cabeçalho de cada pacote e o modelo baseia-se na captura de trinta minutos de dados para processamento posterior.

No estudo feito por [LLS] o problema foi dividido em duas partes. Na primeira fase foi identificado o tráfego correspondente ao protocolo de mensagens IRC[8], na segunda fase foi identificado qual do tráfego IRC corresponde a tráfego gerado por uma *botnet* que usa este tipo de protocolo para difundir as suas mensagens. Neste estudo usaram classificadores *Näive Bayes*, *C4.5* e Redes Baesianas. Para a primeira fase tiveram melhores resultados com *Näive Bayes*, na segunda fase tiveram melhores resultados com a Rede Baeysiana. Este estudo teve uma taxa elevada de falsos positivos (30-40%) e encontra-se desatualizado, atualmente já se observa a um uso inferior de canais IRC para C&C, o que inviabiliza em parte o uso da solução no contexto atual.

Em [ZTSLSGG13] é apresentado um modelo de classificação recorrendo a árvores de decisão, sendo que definiram uma janela de captura variável. O estudo foi feito sobre um conjunto de dados considerável (1,672,575 fluxos de rede) e com resultados de precisão acima de 90% para uma taxa de falsos positivos inferior a 5%. A análise ao tráfego de rede foi feito sobre TCP e UDP, para uma janela de tempo variável. Foram identificadas duas situações para a janela de captura de um fluxo, primeiro, se a janela for demasiado

[8] *Internet Relay Chat* - Protocolo de comunicação por mensagens criado nos anos 80 por Jarkko Oikarinen.

curta, pode haver perda de pacotes relevantes para a deteção futura, segundo, se a janela for demasiado longa, pode não ser possível classificar um fluxo até que a janela de decisão correta seja preenchida. Foi observado que para vários protocolos, a troca inicial de pacotes (*handshake*) tende a ser única e segue um comportamento especifico e variável para cada tipo de aplicação maliciosa. Esta abordagem é particularmente sensível a grandes alterações no tráfego da rede, nomeadamente à existência de tráfego que recai sobre protocolos ponto a ponto, de origem benigna e maliciosa.

No estudo feito por [DRP12] foi usado um método de *clustering* para deteção de *botnets* através de uma análise a tráfego rede cifrado e não cifrado. Foi usada a definição tradicional de um fluxo de rede, sendo que captura é feita tendo em conta as opções seguintes: um fluxo contínuo de pacotes transmitidos numa direção até que a direção do tráfego se altere; ou uma janela de 5 minutos seja atingida; ou um novo fluxo seja aberto. Infelizmente devido às limitações do *gateway* da instituição de onde foram capturados os dados, este estudo apenas considera uma janela de ausência de tráfego de 5 minutos. Seria interessante ter resultados de performance e precisão para um período de tempo superior dado que existem aplicações maliciosas que apresentam janelas de comunicação superiores [GPZL08; ZTSLSGG13].

No estudo feito por [LT12] foram usados classificadores como *Näive Bayes* e *K-Nearest Neighbour*. Foram testados cinco tipo de ataques sobre a rede, foi verificado que dos nove atributos considerados, oito demonstraram uma precisão superior a 75% sobre pelo menos um tipo de ataque. O atributo que demonstrou menor eficácia foi o número de endereços de destino.

2.2.4 Deteção por Informação do Domínio

Este tipo de deteção recai sobre as informações de registo do domínio junto dos serviços de consulta a *nameservers*, no âmbito do protocolo DNS.

[SI] identificou uma solução que recai sobre uma correlação espacial do IP, para qual um domínio resolve. No seu trabalho foram usados o Índice de Morgan e o Coeficiente de Geary, com resultados de precisão na ordem dos 97%, em que 3% se deve a falsos positivos. Recorreram ao top de mil domínios do sitio *Alexa* como *corpus* benigno. Para a correlação usaram três atributos distintos sobre o IP a que um domínio resolve: o par latitude e longitude, a posição Universal Transversa de Mercator (UTM) a localização geográfica de acordo com o *standard* MGRS – sistema de coordenadas geográficas usada pelos militares da NATO para determinar localizações geográficas.

No estudo efetuado por [NPA13] foram consideradas variações da aplicação maliciosa e respetivos padrões. Para cada padrão de comportamento rede foi atribuída uma pontuação de forma a que fosse possível efetuar *clustering* sobre os dados. Para a classificação usaram Modelos Ocultos de Markov (HMM). Na sua análise removeu os TLDs dos domínios com recurso a uma lista pública de sufixos [Moz].

Antonakakis, et al. [ADEJ12] apresenta um bom caso de estudo sobre as DGAs e

exemplo de deteção. Foram detetadas ameaças através de um elevado número de *NXDomains*(domínios inexistentes) por parte do mesmo IP. A empresa dispunha de uma solução de sessões centralizado, onde um utilizador podia efetuar em qualquer computador da empresa, foi observado que o comportamento da aplicação maliciosa seguia a sessão do utilizador independentemente do computador. Para a aplicação maliciosa em estudo, foi observado que tinha duas formas de comunicação com o exterior, a primeira através de comunicação ponto a ponto para contactar com outros computadores infetados, caso não fosse possível a aplicação dispunha de um mecanismo de tolerância a falhas e verificava a ligação à Internet através do serviço de pesquisas Bing ou Google, em caso de sucesso ligava-se ao C&C através de nomes de domínios gerados por uma DGA, para obter atualizações.

2.3 Aprendizagem Automática

A aprendizagem automática tem sido usada em vários campos de investigação, inclusive na deteção de *botnets* (secção 2.2). Pretende-se fazer a sua identificação recorrendo a valores das variáveis medidas que apresentem boas estimativas sobre uma família de eventos com uma distribuição desconhecida.

São apresentadas duas abordagens para obtenção de uma maior confiança nos resultados, a primeira sobre a classificação do nome de domínios com recurso a Máquinas de Vetor de Suporte, a segunda recaiu num algoritmo de *clustering* que se baseia num tipo de rede neuronal auto-associativa denominada mapas de Kohonen, também conhecidos por mapas auto-organizáveis.

2.3.1 Máquina de Vector de Suporte

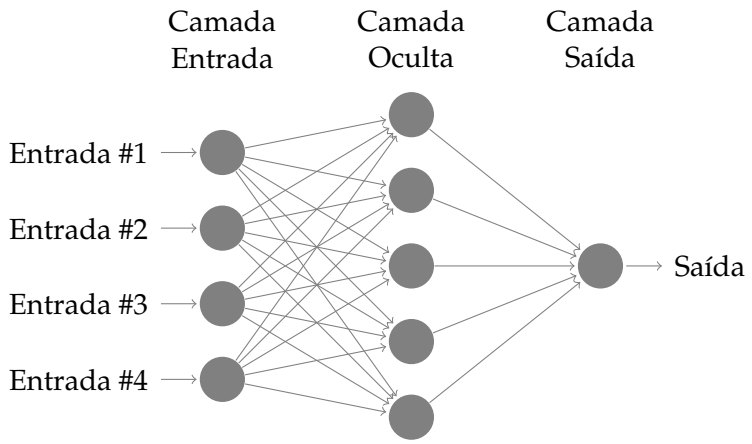
As Máquinas de Vetores de Suporte, adiante designadas de SVM (*Support Vector Machine*), introduzidas por Vladimir N. Vapnik fazem parte da área de aprendizagem estatística. A implementação atual é a proposta por Vladimir Vapnik e Corinna Cortes [CV95], este algoritmo é usado para classificação de atributos e para problemas de regressão, o seu objetivo é encontrar um conjunto de hiper-planos que melhor separe os atributos por classes. Dado que os atributos geralmente não são separáveis, as SVMs introduzem a noção de hiper-plano que separa os dados por espaços de dimensionalidade maior. As SVMs são usadas por exemplo para classificação de e-mail, categorização de sites, reconhecimento ótico de caracteres (OCR) e para ajuda à deteção de *botnets* [DK13]. As SVMs usam uma função de *kernel* escolhida pelo utilizador. Dada uma função f, e os vetores x e y, o *kernel* é uma função em x e y cujo imagem é igual ao produto interno das imagens $f(x)$ e $f(y)$ no D'_f. Dado que $f(x)$ pode ter uma dimensão infinita, basta calcular $< f(x), f(y) >$ usando $K(x, y)$.

$$K(x, y) =< f(x), f(y) > \tag{2.1}$$

O uso de uma função de *kernel* tem a vantagem de permitir transformar o espaço de entrada num espaço com mais dimensões sem ser necessário calcular $f(x)$ porque a SVM só exige o cálculo dos produtos internos.

2.3.2 Redes Neuronais

As redes neuronais artificiais são baseadas nos métodos usados por mecanismos de aprendizagem biológicos [Mit97]. Para cada aplicação possível destaca-se o seu uso para reconhecimento de texto, reconhecimento de voz, reconhecimento de imagens, condução de veículos e deteção de intrusões em sistemas informáticos [AJS06]. As redes neuronais artificiais baseiam-se em quatro pressupostos observados através de modelos matemáticos que reproduzem o pensamento humano ou modelos biológicos [Fau94]: a informação é processada em elementos individuais chamados de neurónios, a informação entre cada neurónio é transmitida através de canais, cada canal tem um peso associado que multiplica o valor da informação que o cruza e cada neurónio aplica uma função de ativação sobre a soma dos canais de entrada para calcular o valor de saída. Encontram-se dispostas por camadas e podem apresentar-se em uni-camada, multi-camada e por camadas concorrentes. A disposição por uni-camada apresenta uma única camada que dispõe de pesos associados a cada ligação, a disposição por multi-camada apresenta duas ou mais camadas. Para cada número de camadas superior a 2, o número de camadas ocultas é N-2, sendo N o número total de camadas existentes na rede neuronal (como exemplificado no diagrama em baixo). A disposição por camada concorrente consiste na competição entre a resposta mais válida, por parte dos neurónios das camadas ocultas.



As redes neuronais são baseadas no algoritmo de *backpropagation* (propagação inversa), a rede neuronal é ativada, no sentido direto, usando os pesos correntes para cada neurónio, depois o erro nos valores de saída é calculado e propagado, no sentido inverso, alterando os pesos do neurónio. A fase de treino termina quando a soma dos erros da rede neuronal nos neurónios de saída é igual, ou menor ao pretendido, ou quando o número de iterações máximas pelo algoritmo de *backpropagation* é atingido. Os pesos em

cada camada podem ser atribuídos através de três formas: treino supervisionado, associação não supervisionada ou através da atribuição de pesos fixos, sem haver necessidade de qualquer tipo de treino. O treino supervisionado recai sobre uma fase de treino que antecede à fase de classificação. Este tipo de treino necessita de uma classificação dos dados de treino, parte-se do principio que existem dados disponíveis para todas as classes de saída da rede neuronal.

2.4 Clustering

O inicio da análise de *clusters*, também denominado de *clustering*, foi iniciado no ano de 1932 sobre a a área de antropologia por Driver e Kroeber [DK32]. O *clustering* é a ação de agrupar objetos com características semelhantes em dois ou mais grupos, sendo atualmente usado usado em vários campos de investigação onde se inclui a aprendizagem automática, reconhecimento de padrões e bio-informática.

Para as técnicas de *clustering* existentes, destacam-se os modelos de clustering por conectividade, *clustering* por centróides, clustering por modelos de distribuição e densidade, e *clustering* por modelos baseados em grafos.

Para esta dissertação foi a escolha recaiu sobre os Mapas de Kohonen, um tipo de rede neuronal auto-associativa desenvolvida por Teuvo Kohonen em 1982.

2.4.1 Redes Neuronais Auto-associativas

Este tipo de Redes Neuronais, proposto por Kohonen [Koh98] diferem das associativas no tipo de arquitetura, para este sub-tipo de redes o vetor de valores de treino é idêntico ao de saída. Este tipo de redes é usado para identificar padrões através de uma entrada com ruído, ou com valores parcialmente próximos de um padrão conhecido. Este tipo de redes memoriza os padrões observados nos valores de entrada, pelo que, e relativamente ao tema em estudo, este tipo de redes é útil para comprimir os atributos aos valores usados na camada intermédia da rede, o que facilita posteriormente a análise dos *clusters* formados.

2.4.2 Mapas Auto-organizáveis

Os Mapas de Kohonen, adiante referidos de SOM (*Self-organizing maps*) e também descritos como mapa auto-organizável, são um tipo de Rede Neuronal Auto-associativa desenvolvido por Teuvo Kohonen em 1982 [Koh82]. Este algoritmo tem uma grelha de saída, também denominada de camada de Kohonen, onde os elementos são inseridos de acordo com uma medida de distância.

Relativamente ao problema em questão, o uso de SOMs tem a desvantagem de necessitar de ter diversos parâmetros à *priori*, incluindo a topologia da própria rede, o que impossibilita o seu uso num método contínuo de classificação. Por esta razão são propostas estruturas adicionais que permitem ter este último ponto em consideração, como é o

caso dos Mapas Auto-organizáveis Crescentes, adiante denominados de GSOM(*Growing Self-organizing Maps*), que permitem um treino *on-line* sem haver de paragem do ciclo de treino. Os GSOM são a base dos mapas auto-organizáveis crescentes hierárquicos; os GHSOMs (Growing Hierarchical Self-organizing Maps) propostos por Rauber, et al. [RMD02] crescem de forma hierárquica a fim de encaixar variações mais pormenorizadas do um conjunto de atributos, nas camadas de percetrões existentes. Ippoliti, et al. [IZ12] propõe uma alteração ao modelo anterior que permite ter em conta variações no número de sub-layers existentes, desta forma é possível criar uma topologia ideal em vez de uma que diminua o erro com as dimensões disponíveis e definidas à *priori*. Palomo, et al. [PLLDL10] propõe uma solução para deteção de anomalias com um algoritmo chamado GHSOM-1. Este algoritmo diminui os parâmetros necessários para a obtenção de resultados através de uma análise às relações hierárquicas dos dados de entrada durante a fase de treino.

3

Contribuições

Este capítulo aborda as contribuições efetuadas para a obtenção de resultados, na primeira secção é feita uma introdução à arquitetura geral da metodologia usada e respetivas tecnologias, a segunda secção descreve o funcionamento do CHEWER, a terceira secção descreve o DNS Crawler, a quarta secção descreve o funcionamento e implementação do CONDENSER, e a última secção descreve o modo de funcionamento do *Clusterer*.

3.1 Arquitetura e Tecnologias

Para atingir os objetivos propostos foram desenvolvidos três sistemas: o CONDENSER - sistema de execução sequencial que extraí informação de várias fontes, efetua *clustering* através do *Clusterer* e classifica nomes de domínio, e guarda a informação numa base de dados orientada a grafos; o DNS Crawler - sistema de histórico de informação DNS sobre nomes de domínio, que faz parte de um módulo do CONDENSER; e o CHEWER - sistema de extração, refinação e classificação de um conjunto de dados de entrada, este sistema é usado para obter o melhor modelo de classificação disponível com o menor conjunto de dados e atributos, por métodos de *bootstrapping* e de seleção de atributos.

A implementação do CONDENSER e do DNS Crawler foi feita com recurso à linguagem de *scripting* Javascript, a base de dados orientada a grafos é uma instância do neo4j, por ter uma comunidade e documentação acessíveis o que levam a que a sua curva de aprendizagem e tempo de implementação sejam reduzidos. Pesquisas que envolvam cálculo de topologias e sub-topologias das comunicações observadas tornam-se mais eficientes com o uso deste tipo de base de dados.

O CHEWER foi desenvolvido em Ruby, e está assente sobre uma versão paralelizada da *LIBSVM* que faz uso do OpenMP, uma API em C de processamento paralelo. Dado

que o treino da SVM está limitada pelo CPU disponível (*cpu-bound*) esta foi uma solução ideal, os treinos foram efetuados numa máquina com o sistema operativo Ubuntu 64 *bits*, com 24 *threads* lógicos de processamento e 32Gb de memória RAM.

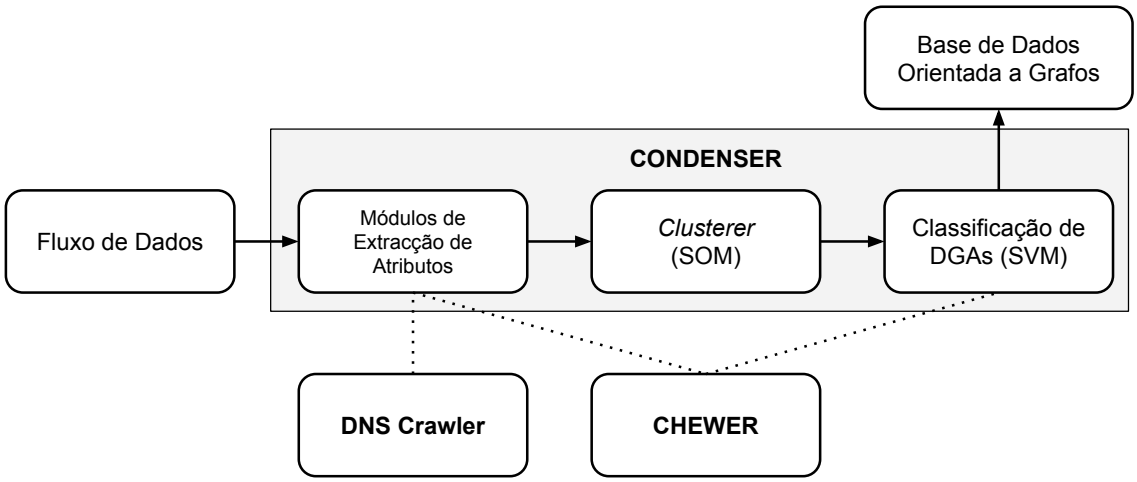


Figura 3.1: Arquitetura da Implementação

A base da implementação segue um fluxo continuo de execução, com passagem por diversos módulos, e que resultam numa inserção numa base de dados orientada a grafos. Na Figura 3.1 encontram-se descritas as dependências de todo o processo, por motivos de clarificação, o sistema CHEWER é usado para determinar algumas medidas de performance tendo em conta o conjunto de dados e respetivos atributos disponíveis para o treino da SVM.

3.2 CONDENSER

O sistema CONDENSER é o nome do processo de redução de dimensionalidade dos resultados obtidos através dos módulos descritos nos sub-capítulos seguintes. Pretende-se agrupar os atributos de acordo com os resultados observados em cada módulo através de um processo de *clustering* e de classificação numa base de dados de grafos. Este sistema encontra-se dividido em diversos componentes.

A Fonte de Dados trata-se de uma componente que efetua uma ligação a uma *feed* que terá ativo o módulo de URI para processamento de URIs, o módulo de GeoIP para geolocalização dos IPs, o sub-módulo Learn para extração de atributos para o classificador de DGAs e de URIs e o sub-módulo de meta informação do sistema Maltracker. Os CONDENSER *Worker*s tratam-se de um componente dividido em vários processos que estão responsáveis por receber trabalho do mestre e por enriquecer a informação recebida através da interação de serviços externos, esta interação é feita através dos seus módulos. A *Cache* Distribuída trata-se de um sistema de *cache* que persiste os dados por um período de tempo pré-determinado, por omissão este valor encontra-se fixo em uma hora.

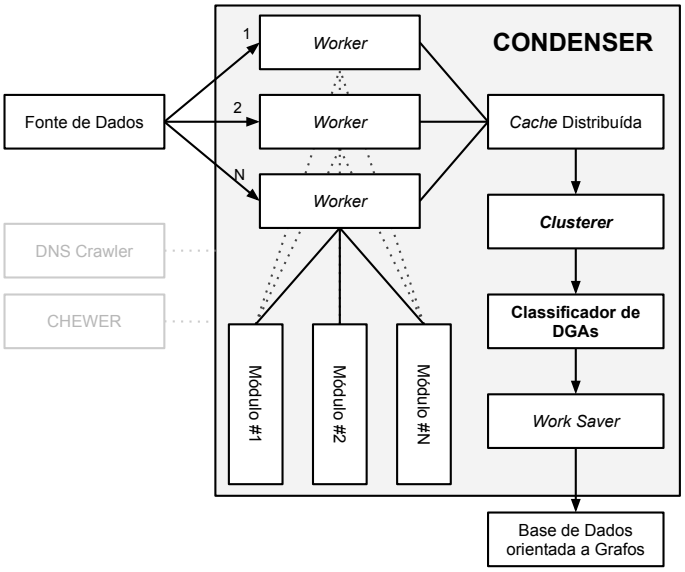


Figura 3.2: arquitetura do CONDENSER

Os *Work Saver*s tratam-se de processos que inserem a informação recolhida nos componentes anteriores numa Base de Dados orientada a Grafos. Este tipo de base de dados é otimizada para algoritmos usados em teoria de grafos, característica que vai ser útil durante a pesquisa de grafos conexos, com o uso de algoritmos que recorrem a pesquisas por profundidade.

3.2.1 Módulos Existentes

Os Módulos do sistema CONDENSER tratam-se de componentes do sistema que fazem a ligação do sistema a serviços externos, nesta secção serão descritos os módulos existentes para uso nos *workers*. O uso de cada módulo é opcional, sendo que mediante o tipo de *feed* disponível pode justificar ativar só parte dos módulos existentes. Os módulos disponíveis no sistema CONDENSER são: o módulo de *queries* ao *DNS Crawler*; o módulo de extração de atributos para o classificador de DGAs e para a fase de *clustering*; o módulo de *queries* aos serviços de reputação IP dos sistemas *Mailspike* e *Spamhaus*; módulo de *queries* à API do *Maltracker* (uma plataforma de análise de aplicações maliciosas proprietária da AnubisNetworks) e um módulo de *queries* a uma API de IPs observados a comunicar com os *sinkholes* da AnubisNetworks. Faz-se de seguida uma descrição mais pormenorizada de cada módulo existente no CONDENSER.

O módulo de interação com o DNS Crawler abstraí a comunicação entre o CONDENSER e o DNS Crawler, sendo feita uma *query* para obter informação relativa aos domínios observados na fonte de dados. Esta componente permite estabelecer relações com IPs de C&Cs conhecidos ou vistos previamente como pertencentes a algum *cluster* suspeito. As *queries* efetuadas ocorrem sobre os nomes de domínio destino e sobre os nomes de domínio do campo *HTTP referer* no caso de se tratar de uma comunicação através do protocolo HTTP.

O módulo de classificação de DGAs, identifica nomes de domínio que aparentem ter origem em DGAs, tipicamente usadas por aplicações maliciosas com recurso a um classificador SVM. Dado que os atributos são extraídos com recurso a um módulo da *feed* da AnubisNetworks, desenvolvido para este efeito (não confundir com os módulos do CONDENSER), os atributos são extraídos diretamente do JSON e, de seguida, os domínios são classificados com intermédio a um modelo do classificador de nomes de domínio gerado pelo CHEWER.

O módulo de interação com o *Maltracker* abstrai a comunicação entre o CONDENSER e a API do Maltracker. Conforme referido anteriormente, o Maltracker é uma plataforma de análise de comportamento de aplicações maliciosas, onde se inclui a comunicação rede efetuada pelas mesmas. Pretende-se detetar domínios e IPs usados para contacto por aplicações maliciosas previamente analisadas, fazendo assim atribuição de aplicações maliciosas aos *clusters* detetados.

O módulo de interação com a API dos *sinkholes* da AnubisNtworks indicam se um IP foi observado a comunicar com um dos *sinkholes* da AnubisNetworks. Presume-se que esse IP está infetado com uma ou mais aplicações maliciosas.

3.3 CHEWER

Foi desenvolvido um programa, designado de CHEWER, que segue um procedimento de execução fixo durante o treino dos classificadores, este programa prepara o conjunto de dados para treino da SVM e permite ações diversas durante e após a fase de treino. A medição de performance dos classificadores também é efetuada de forma automática para todas as funcionalidades que envolvam treino da SVM.

O CHEWER implementa um mecanismo de remoção de dados duplicados do conjunto de dados original, remoção de dados duplicados na intersecção das classes do conjunto de dados original e ordenação aleatória dos dados de entrada. Estas operações permitem refinar os dados de entrada ao remover ruído associado pela presença de dados desnecessários no conjunto de dados.

Está incluída uma chamada para um programa (em JAVA) externo ao CHEWER que extraí os atributos de entrada do conjunto de dados completo, desta forma é possível transpor a operação de extração para qualquer tipo de dados de entrada, sem haver necessidade de alteração do CHEWER de forma direta, com a exceção da indicação do programa a executar nos argumentos de invocação do CHEWER.

Também se encontra disponível uma funcionalidade de normalização, esta opção permite obter uma classificação mais precisa no caso de haver um grande número de ocorrências de atributos que saem dos valores habituais (-1 e 1).

Está disponível uma funcionalidade que permite a execução do treino da SVM com um conjunto de dados variável, que cria partições do conjunto de dados completo, seguindo uma função exponencial e linear para o cálculo da sua dimensão. Antes da execução desta funcionalidade, é removido 10% do conjunto de dados completo com o objetivo

de testar o conjunto de dados após o classificador ser treinado com validação cruzada.

A base da fórmula usada para a criação da partição com recurso à função exponencial, corresponde ao valor exponencial, enquadrado nos limites do conjunto de dados existente.

$$f(i,S) = \frac{2^i \cdot S}{2^{15}} \quad : \quad i,S \in \mathbb{N} \wedge i \in [1,15]$$

Figura 3.3: Formula de Criação dos Sub-Conjuntos de Dados com Dimensão Exponencial

A base da fórmula usada para a criação da partição com recurso à função linear, corresponde ao valor fracionário simples (linear), enquadrado nos limites do conjunto de dados existente.

$$f(i,S) = \frac{i \cdot S}{10} \quad : \quad i,S \in \mathbb{N} \wedge i \in [1,10]$$

Figura 3.4: Formula de Criação dos Sub-Conjuntos de Dados com Dimensão Linear

A SVM é treinada com recurso a uma versão paralelizada da biblioteca LibSVM, que segue os parâmetros de entrada e saída da versão original da implementação de Chih-Chung Chang e Chih-Jen Lin. Após o treino do classificador é feito um cálculo da performance final, usando um sub-conjunto de dados para teste criado após a fase de extração de atributos.

3.4 DNS Crawler

Este sistema traduz-se num conjunto de componentes que efetua análises a domínios observados a fim de detetar comunicações com domínios contendo características potencialmente anómalas.

Este sistema permite identificar parte das técnicas de evasão à deteção, onde se inclui deteção de *fast-flux*, *double fast-flux* e pedidos que originaram respostas `NXDOMAIN`. Dado que os IPs observados nas informações DNS dos Nomes de domínio não são removidas da base de dados, pode-se afirmar que se dispõe de um histórico de informação, havendo desta forma possibilidade de observar as alterações de informação dos domínios ao longo do tempo, e da mesma forma, a rotação de domínios e IPs associados a *botnets*.

O Sistema DNS *Crawler* contêm vários componentes a fim de poder ter uma execução distribuída e escalável. A fonte de dados provem de uma captura no momento, de uma reprodução de uma captura passada ou através de uma *feed* de dados. A componente de ligação à *feed* e comunicação com os *workers* tem um comportamento mestre e está responsável por distribuir informação a ser processada. A componente *Job Sender Worker* está responsável por retirar o domínio do pacote recebido e por inserir o seu valor na *cache* para posterior análise (caso ainda não tenha sido analisado previamente ou não disponha de informação expirada). A *cache* distribuída trata-se de um sistema de *cache*

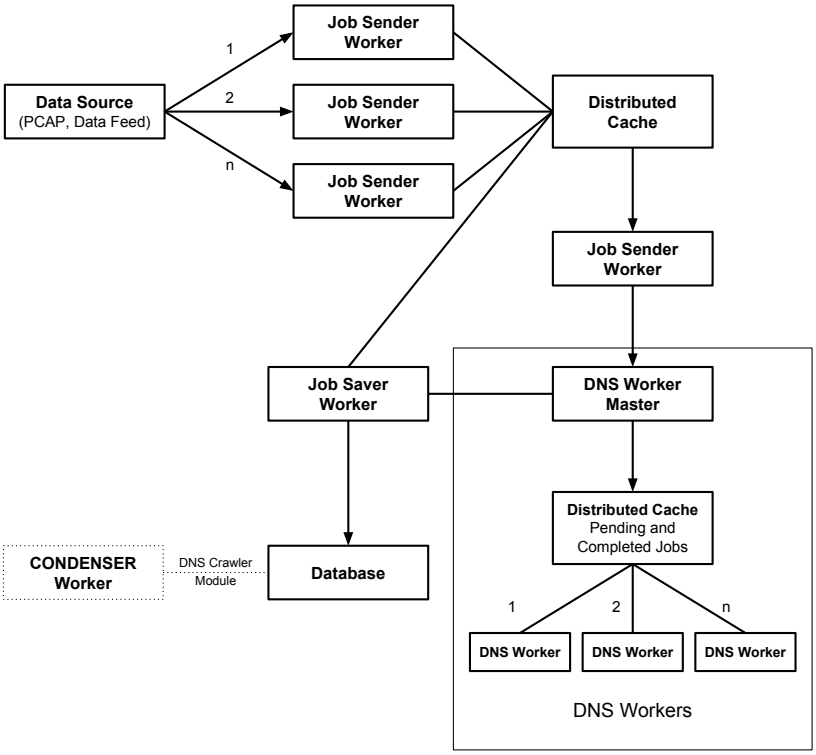


Figura 3.5: Descrição visual das dependências do DNS *Crawler*.

que persiste os dados por um período de tempo pré-determinado, por omissão este valor encontra-se fixo em uma hora. A componente *Job Submiter* trata-se de um sistema de cache que submete os dados para a componente DNS *Worker Master*. Esta componente está responsável por receber trabalhos de resolução DNS e por submeter a uma fila para posterior processamento. Este módulo também aceita trabalhos prioritários, para resolução de nomes de domínio suspeitos de *fast-flux* e *double fast-flux*. Os *DNS Worker*s tratam-se de uma componente que obtém trabalhos submetidos na fila de trabalho e tenta executá-los, após a execução insere os trabalhos numa fila de trabalhos concluídos. A componente *Completed Jobs Saver* obtém os trabalhos concluídos e mediante o resultado pode voltar a submeter o trabalho de forma prioritária. Caso um domínio seja observado com um TTL reduzido é inserido na *cache* como ainda não tendo sido submetido, caso o domínio seja observado com um TTL reduzido e com alterações desde a última resolução é submetido de forma prioritária no DNS *Worker Master*.

3.5 *Clusterer*

A metodologia descrita para o funcionamento do CONDENSER termina no *Clusterer*. Esta componente efetua *clustering* (com recurso a um Mapa de Kohonen) da informação extraída anteriormente e insere o resultado obtido na base de dados, correlacionando informação de histórico obtida através do *DNS Crawler* com os *clusters* formados através do algoritmo de *clustering* usado.

Na Listagem 3.1 encontra-se um exemplo de *output* do *CONDENSER*, antes da inserção na base de dados e sem enriquecimento de informação por parte do DNS Crawler. Neste caso é reproduzido um caso real com alterações de alguns campos. No excerto de JSON na referida Listagem encontra-se demonstrada uma comunicação feita por uma determinada família de aplicações maliciosas, que foi agrupada através de *clustering*. A informação disposta foi alterada nos valores dos IPs e domínios observados por motivos de sigilo da informação.

No exemplo apresentado o campo `hits` refere-se ao número de vezes que foi visto uma comunicação de um certo IP com um domínio, o campo *intersections* refere o número de domínios únicos presentes no mesmo *cluster* com que um IP foi visto comunicar. O campo `seems_dga` é o resultado da aplicação do classificador de nomes de domínio aos domínios presentes no vetor `patterns`. Também são apresentadas métricas adicionais que permitem refinar a informação de cada *cluster* no campo `metrics`, como é o caso do número médio de comunicações por cada IP (`avg_hits`), do número médio de intersecções por cada IP (`avg_intersections`), do número médio de IPs para cada padrão (`avg_ips`), do somatório de acessos de cada IP (`hits`), do quociente entre o número de padrões existentes e do número de IPs (`avg_ip_per_pattern`) e do número de IPs únicos existentes no `cluster`.

```
{
  "ips": {
    "extended": [{
        "hits": 4,
        "intersections": 1,
        "ip": "127.0.0.1"
      },
      {
        "hits": 5,
        "intersections": 1,
        "ip": "127.0.0.2"
      }],
    "values": ["127.0.0.1","127.0.0.2"]
  },
  "metrics": {
    "avg_hits": 4.5,
    "avg_intersections": 1.0,
    "avg_ips": 1.0,
    "hits": 9,
    "avg_ip_per_pattern": 1.0,
    "ips": 2
  },
  "patterns": [{
      "hits": 4,
      "value": {
        "host": "jhia2iu6skja9.com",
        "httpcode": 200,
        "httpversion": "HTTP/1.0",
        "method": "GET",
        "size": 1,
        "uri_path": "/update",
        "seems_dga": true
      }
    },{
      "hits": 5,
      "value": {
        "host": "jsia5iueseja0.com",
        "httpcode": 200,
        "httpversion": "HTTP/1.0",
        "method": "GET",
        "size": 1,
        "uri_path": "/update",
        "seems_dga": true
      }}]
}
```

Listagem 3.1: Resultado de um *cluster* obtido através da aplicação do algoritmo de *clustering* em dados reais, em que que os valores dos atributos e centróides, são abstraídos do utilizador e onde são obtidas métricas úteis, para posterior análise do *cluster*.

4

Resultados

A avaliação dos resultados é feita sobre os dois classificadores disponíveis e sobre o algoritmo de *clustering* usado. Para o classificador de nomes de domínio o conjunto de dados pertencente à classe de domínios *não gerados por DGAs* provêm dos 20.000 domínios com mais atividade do dia 7 de Fevereiro de 2014, obtidos no site *Alexa*. Relativamente aos nomes de domínio não pertencentes à classe foi usado um conjunto de domínios provenientes de vários tipos de DGAs registadas pela AnubisNetworks, contendo um total de 27.000 domínios. Por não se tratar de um conjunto de dados público, foi feita uma análise a nível de distribuição de valores dos atributos do conjunto de dados em uso. Para treino do classificador de URIs, o conjunto de dados pertencentes à classe de URIs sem proveniência maliciosa provem de tráfego gerado sobre o acesso ao top de 500 domínios usados para o classificador de nomes de domínio, num total de 10.000 entradas. Para o conjunto de dados não pertencentes à classe, foi usado 10.000 pacotes de tráfego do conjunto de *botnets* seguidas pela AnubisNetworks. Para a avaliação do algoritmo de *clustering*, foi usado um conjunto de dados, não classificado, e sem qualquer tipo de tratamento, correspondendo a 10.000 pacotes obtidos no dia 9 de Agosto de 2014. Foi também efetuado um cálculo do tamanho do conjunto de dados necessário a fim de diminuir o tempo de treino e classificação dos classificadores, assim como uma avaliação ao número de iterações e taxa de aprendizagem necessários de forma a diminuir o tempo de execução do algoritmo de *clustering* sem haver degradação significativa dos resultados. Nas seguintes secções faz-se uma análise da distribuição dos valores após a extração dos atributos, é feita uma análise à performance dos classificadores e uma análise à performance do algoritmo de *clustering* usado através da variação de parâmetros durante a fase de aprendizagem.

4.1 Atributos Usados

Tendo em consideração o trabalho relacionado e alguns tipos de ficheiros comuns associados à disseminação de aplicações maliciosas, foram escolhidos atributos que incidem sobre a análise de um nome de domínio, do respetivo URI dos pedidos, a extensão dos URIs, o respetivo código de resposta HTTP e a versão do protocolo HTTP usado. Nas secções seguintes estão descritos um total de 70 atributos, dos quais 19 são numéricos e 51 categóricos.

4.1.1 Atributos do Nomes de domínio

Para os atributos sobre os nomes de domínio foram escolhidos, em parte, atributos existentes no trabalho relacionado. Para compreensão na obtenção dos atributos descritos, os atributos correspondentes ao rácio de consoantes, rácio de vogais e rácio de outros caracteres - caracteres que não são vogais, consoantes ou dígitos - é feito através da equação 4.1. Para o rácio de vogais em relação a consoantes é feito através da fórmula 4.2.

Tabela 4.1: Atributos de Nomes de domínio

Atributo	Tipo	Eq.	Descrição
`consonantRatio`	Numérico	4.1	Rácio de Consoantes
`consonantVocalRatio`	Numérico	4.2	Rácio de Consoantes em Relação a Vogais
`domainLength`	Numérico	-	Comprimento do Nome de domínio em relação ao máximo definido no RFC 3986 (255 caracteres)
`othersRatio`	Numérico	4.1	Número de outros caracteres
`vocalRatio`	Numérico	4.1	Rácio de Vogais
`digitRatio`	Numérico	4.1	Rácio de Digitos
`noRepeatsByUniGram`	Numérico	4.3	Número de Repetições por Unigramas
`noRepeatsByBiGram`	Numérico	4.3	Número de Repetições por Bigramas
`noRepeatsByTriGram`	Numérico	4.3	Número de Repetições por Trigramas
`tetraGramAnalysis`	Numérico	4.3	Número de Repetições por Tetragramas
`lowFrequenceOccurrence`	Categórico	-	Ocorrência de um domínio que inicia e acaba com um digito[1]

Para os atributos correspondentes a análises por n-gramas é feita através da Equação 4.3, que faz o quociente entre o número de ocorrências com o número máximo de nGramas possíveis para o conjunto de caracteres em análise.

$$\frac{tipoCaracter}{comprimentoDom} \quad (4.1) \qquad \frac{consoantes - vogais}{consoantes + vogais} \quad (4.2)$$

Para os atributos correspondentes ao número de repetições por uni, bi, tri e tetragramas é feito um cálculo da repetição das vogais ou consoantes em sequência através do referido número de caracteres. Para melhor perceção do significado de cada atributo, são apresentados na Tabela B.4 os atributos usados neste classificador.

$$\frac{ocorrenciasPornGramas}{comprimentodomnio - (nGram - 1)} \quad (4.3)$$

Para o atributo `lowFrequenceOccurrence` é feita uma análise ao carácter inicial e final do 2LD com o objetivo de determinar se o domínio faz parte de um domínio com pouca ocorrência. Para o uso deste atributo foi considerada a informação disponível em [Dat12] na qual foi feita uma análise a todos os domínios registados nas áreas *.com* e *.net*, onde foi observado que os domínios que começam e terminam com dígitos mostram pouca probabilidade de ocorrência, com este atributo pretende-se perceber se existe uma relação direta da baixa probabilidade de ocorrência com a geração de `NXDOMAINS` que obedeçam a esta regra.

4.1.2 Atributos de URIs

Os atributos dispostos na Tabela 4.2 são relativos a atributos com finalidade de separar de forma simples a ocorrência de URIs, os atributos estendidos pretendem complementar esta informação e separar os URIs por tipo e estrutura de extensão. Os atributos correspondentes a atributos numéricos são normalizados com o máximo observado no conjunto de dados usado para treino.

Relativamente ao classificador de URIs, foram usados os atributos descritos nesta secção com a exceção do atributo `uriExistence`. Todos restantes atributos quer nesta e nas próximas secções relativas a atributos de URIs *estendidos* foram usados apenas para o algoritmo de *clustering*.

Para a normalização destes atributos é usada a Equação 4.4 que usa o valor máximo observado após a extração do atributo.

$$\frac{occurrenceNumber}{maxOcurrenceNumber} \quad (4.4)$$

4.1.3 Atributos de URIs ao Nível da Extenção

Estes atributos serão usados adicionalmente aos restantes atributos de URIs durante a fase de *clustering*. Pretende-se separar os URIs por tipo de extensão, sendo que a análise

Tabela 4.2: Atributos de URIs

Atributo	Tipo	Eq.	Descrição
`queryLength`	Numérico	4.1	Comprimento do URI
`queryArgumentSize`	Numérico	4.1	Número de Argumentos
`uriBaseLength`	Numérico	4.1	Comprimento da propriedade *uri base*
`uriBaseLevelLength`	Numérico	4.1	Comprimento do número de níveis da propriedade *uri base*
`uriPathLength`	Numérico	4.1	Comprimento da propriedade *uri path*
`uriExistence`	Categórico	-	Existência do Campo de URI

estendida ao URI incide apenas sobre análise a atributos adicionais relativos à extensão. No conjunto de atributos disponíveis existem 34 atributos categóricos que correspondem a extensões de ficheiros relevantes no contexto de infeção e disseminação de aplicações maliciosas.

Os restantes atributos pretendem ter em conta a inexistência de extensões e extensões não existentes nas que são dispostas na primeira linha da Tabela 4.3. À semelhança do grupo de atributos anterior, os atributos são normalizados com os máximos observados do conjunto de dados completo.

4.1.4 Outros atributos

Também foram considerados atributos adicionais, nomeadamente o comprimento dos pacotes, o código HTTP de resposta e a versão do protocolo HTTP usado, num total de 13 atributos categóricos. Para determinação dos atributos relativos ao código de resposta HTTP e versão do protocolo HTTP, foi feito uma análise à ocorrência destes atributos na fonte de dados usada, onde se concluiu que as versões apresentadas são as mais comuns. A fim de antever tráfego que não respeita o protocolo HTTP por defeito, ou com valores pouco usuais dentro do protocolo, foram criados adicionalmente os atributos `unknownReplyCode` e `inexistantHttpVersion`.

4.2 Análise Estatistica

Para determinar a forma como os dados estão dispostos no conjunto de dados completo foi feita uma análise estatística aos conjuntos de dados existentes, esta análise é feita para que se compreenda melhor a distribuição estatística do conjunto de dados usado e para que possam ser feitas comparações com conjuntos de dados usados na literatura, ou mesmo para gerar conjuntos de dados semelhantes aos usados nesta dissertação. Os

Tabela 4.3: Atributos de URIs Extendidos

Atributo(s)	Tipo	Eq.	Descrição
`exe, bat, cmd, msi, com, drv, js, css, dat, ppt, doc, docx, txt, rtf, php, cgi, asp, aspx, html, xhtml, jsf, dll, png, jpg, bmp, bin, dll, zip, rar, swf, scr, wpad, pac, ini`	Categórico	4.1	Extensão do Ficheiro
`unknownExtension`	Categórico	4.1	Extensão Desconhecida
`unavailableExtension`	Categórico	4.1	Extensão Inexistente
`consonantRatio`	Numérico	4.1	Rácio de Consoantes
`vocalRatio`	Numérico	4.1	Rácio de Vogais
`consonantVocalRatio`	Numérico	4.1	Rácio de Consoantes e Vogais
`extensionLength`	Numérico	4.1	Comprimento da Extensão

atributos usados para a analise estatística em baixo correspondem ao valor dos atributos após a extração, antes do trino dos classificadores e *clusterer*.

$$\sigma = \sqrt{\sum_{i=1}^{N} p_i (x_i - \mu)^2}, \text{ onde } \mu = \sum_{i=1}^{N} p_i x_i \text{ , e } p_i = \frac{n_i}{N} \tag{4.5}$$

Foram usadas as fórmulas de obtenção clássicas do desvio-padrão, média e média da distribuição. No caso do cálculo do intervalo de confiança – representado por $CI_{0.95}(1 - \alpha)$ – assumiu-se que o conjunto de dados existente apresenta uma distribuição não normal. A fórmula de cálculo do desvio-padrão encontra-se descrita na Equação 4.5, enquanto que a média $\overline{X}$ encontra-se definida na definição 4.6.

$$\overline{X} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{4.6}$$

Em teoria de probabilidades o Teorema do Limite Central 4.7 defende que a média aritmética de uma distribuição de variáveis independentes aleatórias, cada uma com um valor e variância esperados, se aproximará ao de uma distribuição normal para uma dimensão do conjunto de dados superior a trinta.

$$Z = \frac{\overline{X} - \mu}{\sqrt{\sigma^2 / N}} = \sqrt{N} \frac{\overline{X} - \mu}{\sigma} \tag{4.7}$$

Tabela 4.4: Outros atributos

Atributo(s)	Tipo	Eq.	Descrição
`packetSize`	Numérico	-	Tamanho do Pacote TCP/IP
`packetSizeInexistence`	Categórico	-	Existencia do Tamanho do Pacote TCP/IP
`200, 301, 400, 404, 413`	Categórico	-	Código de Resposta HTTP
`unknownReplyCode`	Categórico	-	Código de Resposta HTTP Desconhecido
`inexistentHttpRCode`	Categórico	-	Código de Resposta HTTP Inexistente
`HTTP/1.0, HTTP/1.1`	Categórico	-	Versão HTTP
`unknownHttpVersion`	Categórico	-	Versão HTTP Desconhecida
`inexistentHttpVersion`	Categórico	-	Versão HTTP Inexistente

Assume-se um valor de significância igual a 0.05

$$1 - \alpha = 0.95 \Rightarrow \alpha = 0.05 \Rightarrow 1 - \alpha/2 = 0.975 \tag{4.8}$$

e assim calcula-se o valor da probabilidade inversa acumulada.

$$z_{\alpha/2} = z_{0.025} = \Phi^{-1}(0.975) = 1.96 \tag{4.9}$$

Aplica-se o valor obtido anteriormente e obtém-se a equação do intervalo de confiança, representada na Equação 4.10.

$$IC_{0.95}(1-\alpha) = \left[\overline{X} - \frac{\sigma}{\sqrt{N}} z_{\alpha/2}, \overline{X} + \frac{\sigma}{\sqrt{N}} z_{\alpha/2}\right], \text{ onde } z_{\alpha/2} = 1.96 \tag{4.10}$$

Nas próximas secções são discutidos os resultados obtidos através da aplicação dos das fórmulas em cima, sendo feita uma análise estatística para os dois conjuntos de dados pertencentes às classes disponíveis de cada classificador, no caso do conjunto de dados do *clusterer* é feita uma análise ao conjunto de dados total.

É importante referir que esta análise é superficial e que não reflete de forma direta os valores obtidos para o método de seleção de atributos usado, uma das razões tem que ver com a forma como os algoritmos de aprendizagem automática - neste caso uma máquina de vetores de suporte - cria os vetores de suporte no hiper-plano, que tem em consideração a totalidade dos atributos e não só a análise singular que é feita na Figura 4.1.

4.2.1 Classificador de Nomes de domínios

Para o classificador de nomes de domínios a análise estatística mostrou resultados interessantes na medida que permite antever parcialmente o resultado da seleção de atributos efetuada demonstrada na secção 4.4. Na Tabela B.1 e Tabela B.2 do Apêndice, são apresentados os valores obtidos para a análise estatística efetuada. Na Figura 4.1 é feita uma disposição das médias obtidas para cada atributo, e para cada conjunto de dados. A preto encontra-se representado o desvio-padrão observado. É possível verificar que alguns dos atributos mostram ter uma grande dispersão ao longo do conjunto de dados, como é o caso dos atributos: `consonantRatio`, `consonantVocalRatio`, `vocalRatio`, `digitRatio` e o atributo `tetraGramAnalysis`.

Relativamente às médias observadas, e em comparação com os dois conjunto de dados deste classificador, o atributo `tetraGramAnalysis` mostrou ter a maior diferença em termos médios. É observado que o conjunto de dados contendo os nomes de domínio gerados por DGA apresenta em termos médios um maior número de ocorrência de consoantes em relação ao tamanho total, facto que se verifica de forma inversa relativamente ao rácio de vogais, em que o conjunto de dados de domínios normais apresenta maior ocorrência de vogais. Para o atributo `consonantVocalRatio` o conjunto de dados relativo a amostras de DGAs mostra ter mais consoantes quando comparado com o número de vogais presente no total do conjunto de carateres do domínio. Para os restantes valores é observado que os dois conjuntos de dados apresentam alguma similaridade.

4.2.2 Classificador de URIs

No Gráfico 4.2 está representado o resultado estatístico da análise aos atributos do classificador de URIs. Foi observado que o atributo mais discriminante foi o `tetraGramAnalysis`, tendo o atributo `queryArgumentSize` também uma variância significativa. Para os restantes valores foi observado que não existe grande discrepância entre os conjunto de dados da classe analisada.

Figura 4.1: Gráfico das médias e desvio-padrão de cada atributo nos conjunto de dados usados para o classificador de nomes de domínio

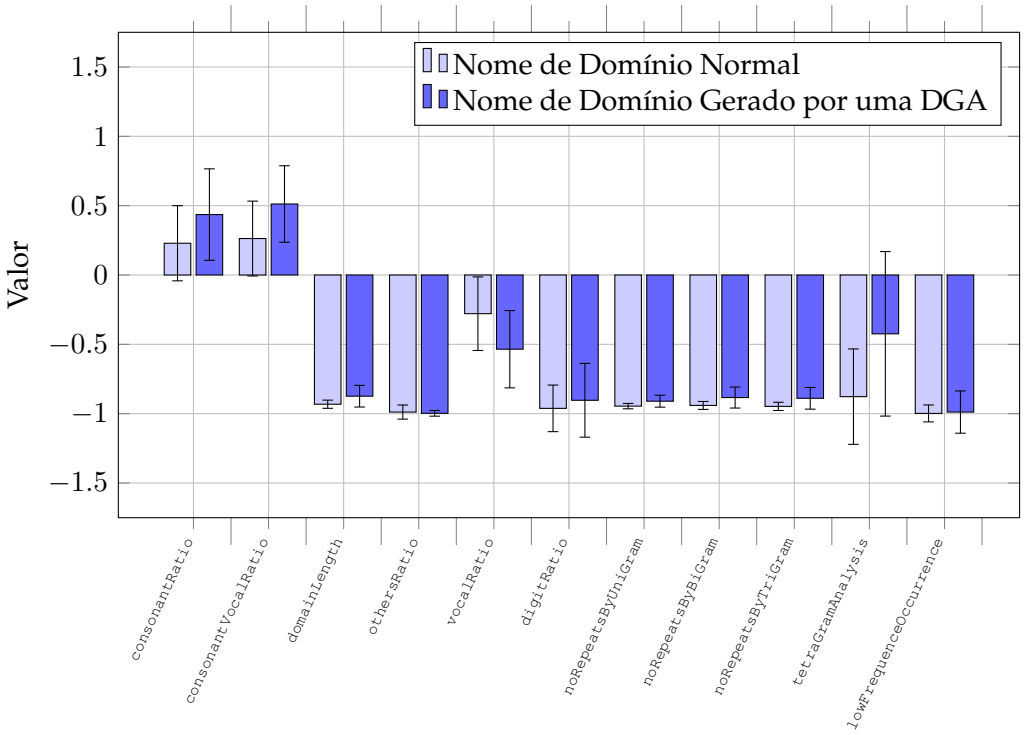
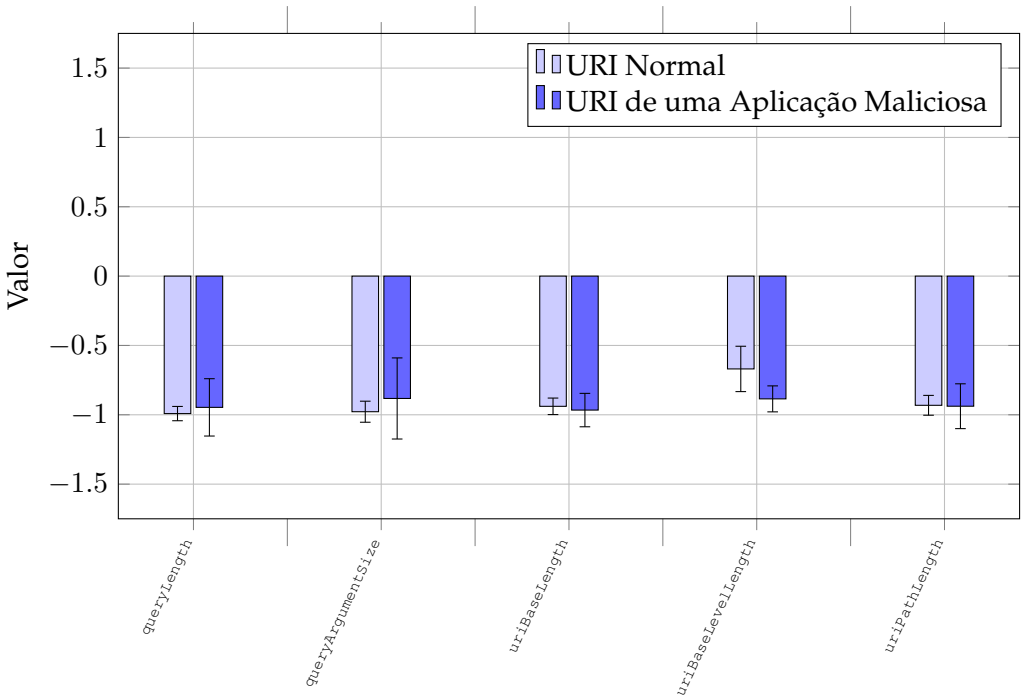


Figura 4.2: Gráfico das médias e desvio-padrão de cada atributo nos conjunto de dados usados para o classificador de URIs



4.3 Avaliação dos Classificadores

Nesta secção discutem-se os resultados obtidos após a avaliação de performance efetuada sobre os dois classificadores. Foram usadas quatro métricas de classificação para a avaliação dos classificadores: *Accuracy*, Precisão, *Recall* e *F-Measure*.

Para esta secção assume-se que tp corresponde ao número de classificações corretas como pertencendo à classe de saída, tn corresponde ao número de classificações corretas como não pertencendo à classe de saída, fp corresponde ao número de classificações incorretas como pertencendo à classe de saída e fn corresponde ao número de classificações incorretas como não pertencendo à classe de saída. O *recall* e precisão são calculados de acordo com as equações 4.11 e 4.12 respetivamente.

$$recall = \frac{tp}{tp + fn} \quad (4.11) \qquad precisão = \frac{tp}{tp + fp} \quad (4.12)$$

A *accuracy* e a *f-measure* são calculadas através das expressões 4.13 e 4.14 respectivamente.

$$accuracy = \frac{tp + tn}{tp + fp + fn + tn} \quad (4.13) \qquad f - measure = \frac{2 * precisão * recall}{precisão + recall} \quad (4.14)$$

Nas próximas secções são apresentados os resultados obtidos para a performance de classificador após o treino com várias dimensões dos conjuntos de dados usado. Para a criação dos sub-conjuntos de dados foram usadas duas funções enquadradas na dimensão total dos conjuntos de dados. No primeiro caso foi usada uma função exponencial com o objetivo de observar o crescimento de performance do classificador com o aumento do tamanho do conjunto de dados, para perceber a partir de que dimensões o classificador começa a apresentar relativa estabilidade. No segundo caso foi usada uma função linear a fim de cobrir as restantes dimensões dos conjuntos de dados.

Foi observado que a equação exponencial demonstrou ter melhores resultados para um período de tempo mais reduzido quando comparada com a equação linear devido às dimensões dos sub-conjuntos de dados usados.

4.3.1 Classificador de Nomes de domínio

Este classificador indica se um domínio é visto como não sendo anómalo, ou por outras palavras, se não aparenta ter sido gerado por nenhum tipo de DGA. Durante a fase de treino com o conjunto de dados total foi atingida uma precisão de 77,9474% para 49.020 vetores de entrada, as restantes métricas são apresentadas na tabela 4.5.

Para o conjunto de treino com dimensão exponencial observou-se que o classificador começa a apresentar estabilidade a partir da sétima iteração. Na Figura 4.3 pode-se observar a evolução da precisão mediante o sub-conjunto de entrada usado. No eixo das

Conjunto Pertencente à Classe	Conjunto não Pertencente à Classe	Accuracy	Precisão	Recall	F-Measure
26.891	27.043	82,0508%	77,9474%	89,2525%	83,2178%

Tabela 4.5: Performance do Classificador de Nomes de domínio com o Conjunto de Dados Completo

ordenadas do lado direito encontra-se disposta a dimensão do sub-conjunto para a iteração disposta no eixo das abcissas. Os valores obtidos e usados para desenho dos gráfico encontram-se em Anexo, na Tabela A.1.

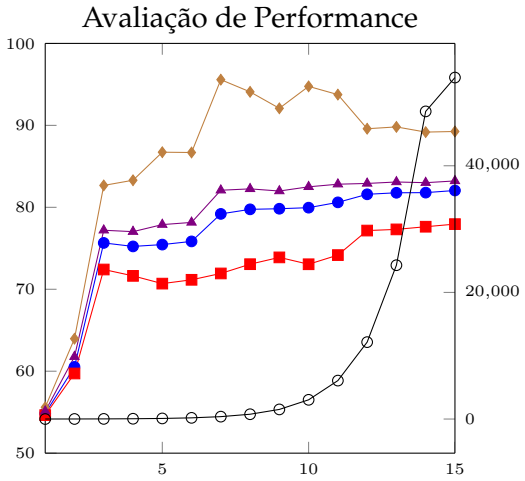


Figura 4.3: Sub-Conjuntos de Dados com Crescimento Exponencial do Classificador de URIs

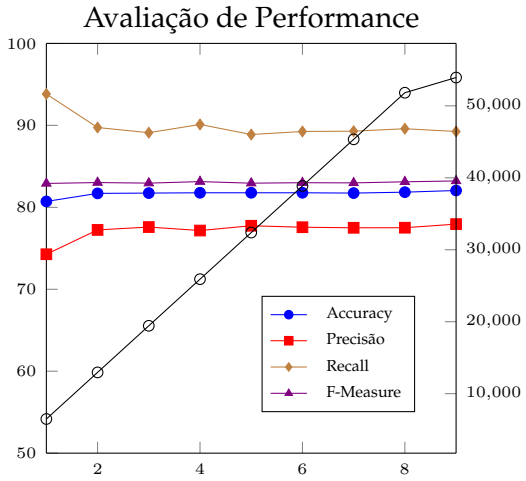


Figura 4.4: Sub-Conjuntos de Dados com Crescimento Linear do Classificador de URIs

Para o conjunto de treino com dimensão linear observou-se que o classificador começa a apresentar estabilidade logo na primeira iteração. Na Figura 4.4 pode-se observar a evolução da precisão mediante o sub-conjunto de entrada usado. No eixo das ordenadas do lado direito encontra-se disposta a dimensão do sub-conjunto para a iteração disposta no eixo das abcissas. Os valores obtidos e usados para desenho dos gráfico encontram-se em Anexo, na Tabela A.2.

4.3.2 Classificador de URIs

Este classificador indica se um domínio é visto como não sendo anómalo, ou por outras palavras, se não aparenta ter sido gerado por nenhum tipo de DGA. Durante a fase de treino com o conjunto de dados total foi atingida uma precisão de 77,9474 para 49.020 vetores de entrada, as restantes métricas são apresentadas na tabela 4.6.

Para o conjunto de treino com dimensão exponencial observou-se que o classificador começa a apresentar estabilidade a partir da sétima iteração. Na Figura 4.5 pode-se observar a evolução da precisão mediante o sub-conjunto de entrada usado. No eixo das

Conjunto Pertencente à Classe	Conjunto não Pertencente à Classe	Accuracy	Precisão	Recall	F-Measure
9910	9748	96,7430%	96,3037%	97,2755%	96,7872%

Tabela 4.6: Performance do Classificador de URIs com o Conjunto de Dados Completo

ordenadas do lado direito encontra-se disposta a dimensão do sub-conjunto para a iteração disposta no eixo das abcissas. Os valores obtidos e usados para desenho dos gráfico encontram-se em Anexo, na Tabela A.3.

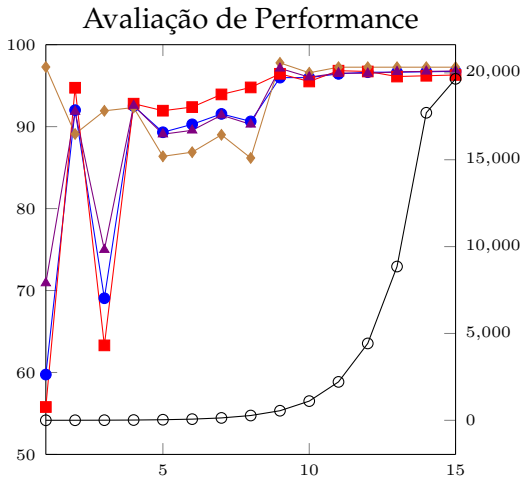


Figura 4.5: Sub-Conjuntos de Dados com Crescimento Exponencial do Classificador de URIs

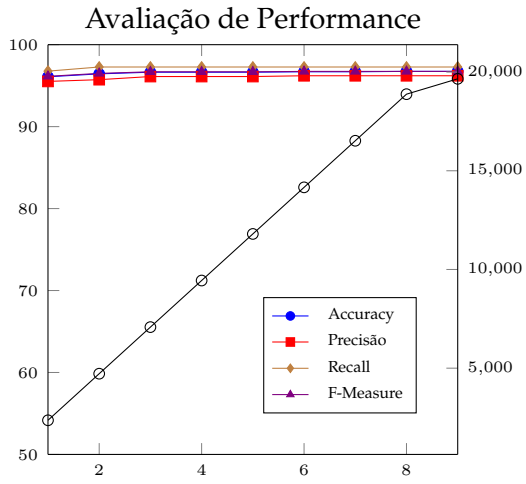


Figura 4.6: Sub-Conjuntos de Dados com Crescimento Linear do Classificador de URIs

Para o conjunto de treino com dimensão linear observou-se que o classificador começa a apresentar estabilidade logo na primeira iteração. Na Figura 4.6 pode-se observar a evolução da precisão mediante o sub-conjunto de entrada usado. No eixo das ordenadas do lado direito encontra-se disposta a dimensão do sub-conjunto para a iteração disposta no eixo das abcissas. Os valores obtidos e usados para desenho dos gráfico encontram-se em Anexo, na Tabela A.4.

4.4 Seleção de Atributos

Para determinar qual o conjunto de atributos mais discriminante do conjunto de dados usado para o classificador de nomes de domínio, foi desenvolvida uma funcionalidade no CHEWER que permite executar treinos de um ou mais subconjuntos do conjunto de atributos completo. Para determinar qual o melhor conjunto de atributos foi seguido o seguinte procedimento:

1. Seleção de atributos para todos os sub-conjuntos dos atributos disponíveis, em conjuntos 2 a 2 até $N-1$ a $N-1$ (em que N é o conjunto total de atributos disponíveis).

Para este passo foi usado um sub-conjunto composto por 250 entradas aleatórias do conjunto de dados completo.

2. Seleção de atributos dos 36 atributos que apresentaram melhor performance na execução anterior. Para este passo foi usado um sub-conjunto composto por 500 entradas aleatórias do conjunto de dados completo.

3. Seleção de atributos do top de 7 atributos que apresentaram melhor performance na execução anterior. Para este passo foi usado um sub-conjunto composto por 1000 entradas aleatórias do conjunto de dados completo.

4. Seleção de atributos do top de 3 atributos que apresentaram melhor performance na execução anterior. Para este passo foi usado um sub-conjunto composto por 4000 entradas aleatórias do conjunto de dados completo.

5. Treino do classificador com o conjunto de atributos que apresentou a melhor performance na execução anterior. Para esta passo foram usados dois sub-conjuntos, o primeiro composto por 4000 entradas e o segundo por 8000 do conjunto de dados completo.

Neste procedimento faz-se uma assunção de que a execução do treino com os sub-conjuntos do conjunto de dados completo, que segue um crescimento exponencial já foi feita, sendo a dimensão do conjunto de dados do primeiro passo a que apresenta uma estabilidade relativa quando comparada com o conjunto de treino completo. Relativamente a este passo, deveria ser calculado um intervalo de confiança por *bootstraping* com o objetivo de arranjar um sub-conjunto de dados representativo do conjunto total.

Tabela 4.7: Selecção de Atributos para o Classificador de Nomes de domínio

Nº Conj. Features	**Precisão Máxima**	**Abributos Selecionados**	**Dimensão**
2035	79,9	`othersRatio, biGramAnalysis, triGramAnalysis, tetraGramAnalysis`	250
36	83,9	`othersRatio, biGramAnalysis, triGramAnalysis, tetraGramAnalysis`	500
7	83,9	`othersRatio, biGramAnalysis, triGramAnalysis, tetraGramAnalysis`	1000
3	80,7	`domainLength, othersRatio, tetraGramAnalysis`	4000
1	82,9	`domainLength, othersRatio, tetraGramAnalysis`	4000
1	82,1	`domainLength, othersRatio, tetraGramAnalysis`	8000

Na Tabela 4.7 encontram-se os resultados obtidos através da seleção de atributos aplicada ao classificador de nomes de domínio. Foi observado que os atributos mais discriminantes são os correspondentes ao comprimento do nome do domínio (`domainLength`), o rácio de outros caracteres (`othersRatio`), e a análise de repetições por tetra-gramas (`tetraGramAnalysis`).

4.5 Avaliação do *Clusterer*

Tendo em conta o contexto desta dissertação é possível usar os algoritmos de *clustering* propostos com dois objetivos. Para o primeiro caso poderá ser usada para agrupar um conjunto de dados e depois analisar cada *cluster* de forma individual com o objetivo de classificação, através do uso do classificador de nomes de domínio ou de ocorrências iguais ou similares observados no *Maltracker* ou fonte de dados de tráfego malicioso, na prática este método assenta sobre um mapa de atributos auto-organizável (*Self Organizing Feature Map*) e permite uma forma autónoma de fazer a classificação dos *clusters formados*. Para o segundo caso, o método de *clustering* proposto apenas servirá para agrupar os dados de entrada, sem classificação prévia, onde haverá necessidade de intervenção de um analista para classificação manual dos dados obtidos.

Para determinar a performance do SOM ou Mapa de Kohonen foi feita uma validação por validação cruzada em cinco conjuntos mutuamente exclusivos.

Na análise de *clusters* existem duas formas de efetuar a validação dos *clusters* criados por algoritmos de *clustering*: por validação externa, que é baseada numa classificação prévia dos dados existentes nos *clusters*; e validação interna, em que a validação é feita sobre os valores obtidos para os centróides de cada *cluster* e o valor dos elementos que fazem parte dos próprios *clusters* [RA11]. Para o primeiro caso não será feita uma análise dado que o método de *clustering* proposto é usado para agrupar os dados e não para classificação, no entanto são apresentadas algumas métricas a titulo informativo. Para o segundo caso é feito o cálculo de duas medidas de performance.

Os indices de avaliação apresentados servem para optimizar os parâmetros de treino, sendo que a avaliação final da ferramenta será feita pelos analistas da empresa, em função da sua utilidade.

Para as métricas apresentadas, assume-se que a medida de distância usada para agrupar um conjunto de atributos correspondente a uma entrada é a distância Euclidiana que é dada pela Equação 4.15.

$$||x - y|| = \left(\sum_{i=1}^{d} (x_i - y_i)^2 \right)^{1/2} \tag{4.15}$$

Para a medida de performance são apresentadas seis métricas, três correspondentes a métodos de validação externos, três correspondentes a métodos de validação internos. Para as medidas de performance de validação externa refere-se a existência do *recall*, cálculo da precisão e a *f-measure*. Para as medidas de performance de validação

interna [SRS07] foi usado o índice de *Davies-Bouldin* (DB) – Equação 4.17 – o índice de *Silhouette* – Equação 4.21 – e o índice de *Dunn* – Equação 4.18.

$$diam(c_i) = \left(\frac{1}{n_i} \sum_{x \in c_i} ||x - z_i||^2 \right)^{1/2} \tag{4.16}$$

O índice de Davies-Bouldin [DB79] – representado por DB_k – identifica *clusters* que são distantes entre si e compactos, na Equação 4.16 é representado o cálculo do diâmetro do *cluster* c_i, em que n_i representa o número de pontos pertencentes ao *cluster* c_i. O símbolo z_i corresponde ao centróide do *cluster* c_i e x é um ponto pertencente ao *cluster* c_i.

$$DB_k = \frac{1}{k} \sum_{i=1}^{k} Max_{j=1,\ldots,k, i \neq j} \left\{ \frac{diam(c_i) + diam(c_j)}{||c_i - c_j||} \right\} \tag{4.17}$$

Para DB_k, k corresponde ao número total de *clusters* existentes, c_i e c_j correspondem aos centróides dos *clusters* i e j respetivamente.

$$DU_k = \min_{i=1,\ldots,k} \left\{ \min_{j=i+1,\ldots,k} \left(\frac{diss(c_i, c_j)}{max_{m=1,\ldots,k} diam(c_m)} \right) \right\} \tag{4.18}$$

O índice de Dunn [Dun74] – representado por DU_k – identifica *clusters* que se encontram bem separados e compactos. O objetivo é portanto de maximizar a distância inter-*cluster* e minimizar a distância intra-*cluster*.

$$diss(c_i, c_j) = min_{x \in c_i, y \in c_j} ||x - y|| \tag{4.19}$$

Na Equação 4.18, k corresponde ao número total de *clusters* existentes, 4.19 corresponde à medida de dissemilhança entre os *clusters* c_i e c_j, 4.20 corresponde ao diâmetro do *cluster* C.

$$diam(C) = max_{x,y \in C} ||x - y|| \tag{4.20}$$

O índice de Silhouette [L. 90] – representado por SI_k – determina a média da pertença de cada ponto a todos os k *clusters*, onde n corresponde ao número total de pontos existentes no conjunto de dados, a_i é a distância média entre o ponto i e todos os outros pontos pertencentes ao seu *cluster*; b_i é o valor mínimo da dissimilaridade média entre o ponto i e todos os outros *clusters*. Nesta medida de avaliação, a partição com o maior valor de SI considera-se ótimo.

$$SI_k = \frac{1}{n} \sum_{i=1}^{n} \frac{(b_i - a_i)}{max(a_i, b_i)} \tag{4.21}$$

A biblioteca usada para o processo de *clustering* foi a *neuroph*. Esta biblioteca permite, para os Mapas de Kohonen, a especificação do número de iterações na primeira e segunda fase, assim como do *learning rate*, adiante designada de taxa de aprendizagem.

Número de *Clusters*	Índice de Dunn	Índice de Davies-bouldin	Índice de Silhouette
840	0.0001	9.3262	0.2685

Tabela 4.8: Performance da SVM para o Conjunto de Dados Completo

Para os atributos descritos e métodos de avaliação, foram obtidos os seguintes resultados com um conjunto de dados contendo 10.000 entradas, onde foram feitas 20.000 iterações para a primeira fase de treino e 10.000 iterações para a segunda fase, com uma taxa de aprendizagem de 0,5.

Para diminuir o tempo de execução do algoritmo de *clustering* foi feita uma análise ao número de iterações necessária em relação à taxa de aprendizagem usada. Primeiro foi executado o algoritmo com o número máximo de iterações, com o valor de 10.000 iterações na primeira fase e de 20.000 iterações na segunda fase para as várias taxas de aprendizagem apresentadas. Para o número mínimo de iterações foi considerado o valor de 2.000 iterações tanto para a primeira como para a segunda fase. Foi observado que uma taxa de aprendizagem reduzida para os níveis de iteração mínimos (2.000) a primeira e segunda taxas de aprendizagem mostraram pouca performance, ao contrário de taxas de aprendizagem mais elevadas (10.000 e 20.000 iterações respectivamente) que apresentaram uma melhor performance a partir do valor 0.25. Na Figura 4.7 pode-se observar que o número máximo de *clusters* formados para o número mínimo de iterações teve o valor de 937 para uma taxa de aprendizagem de valor 0,45; por outro lado o melhor valor máximo observado para os níveis de iteração máximos foi de 937 *clusters* para uma taxa de aprendizagem correspondente a 0,45. Na Figura 4.8 observa-se que o número mínimo obtido para o Índice de Silhouette foi de 0,0001 para uma taxa de aprendizagem de 0.5, nos níveis de iteração máximos; o valor máximo foi de 0,38 para uma taxa de aprendizagem correspondente a 0,4 nos níveis de iteração mínimos. Por fim, na Figura 4.9, verifica-se que o número mínimo obtido para o Índice de Davies-bouldin foi de 7,28 para uma taxa de aprendizagem de 0.25, nos níveis de iteração máximos, e o valor máximo foi de 10,3 para uma taxa de aprendizagem correspondente a 0,3 nos níveis de iteração máximos.

Pode-se concluir que como era esperado, os valores de aprendizagem reduzidos para os níveis de iteração mínimos apresentaram performance reduzida porque o algoritmo ainda não teve capacidade de aprender toda a informação relativa ao conjunto de dados para o valor de iterações usado. Este facto também ajuda a explicar porque é os valores de taxa de aprendizagem mais elevados apresentaram boa performance, havendo mais liberdade para progredir no avanço da aprendizagem a rede neuronal converge mais rapidamente para uma solução.

Figura 4.7: Histograma da dimensão dos *clusters* com variação do número de iterações usadas para treino do mapa auto-organizável.

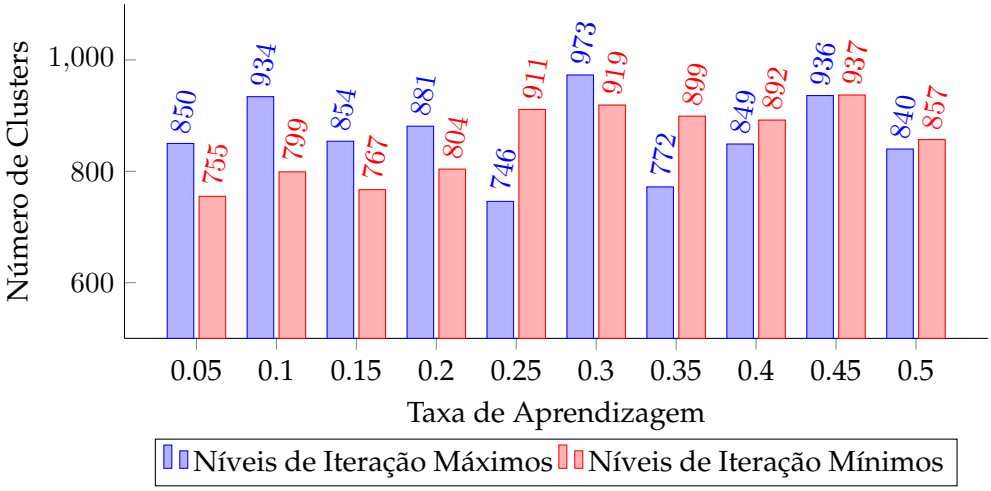
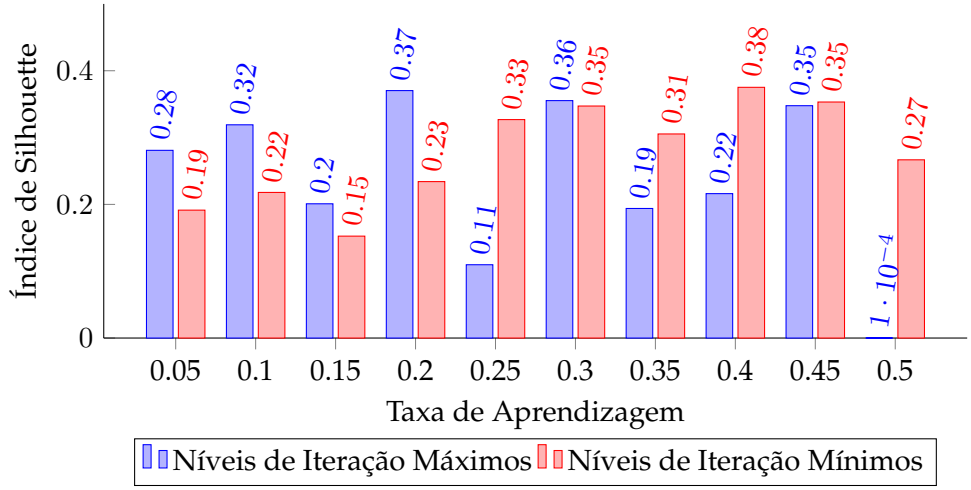


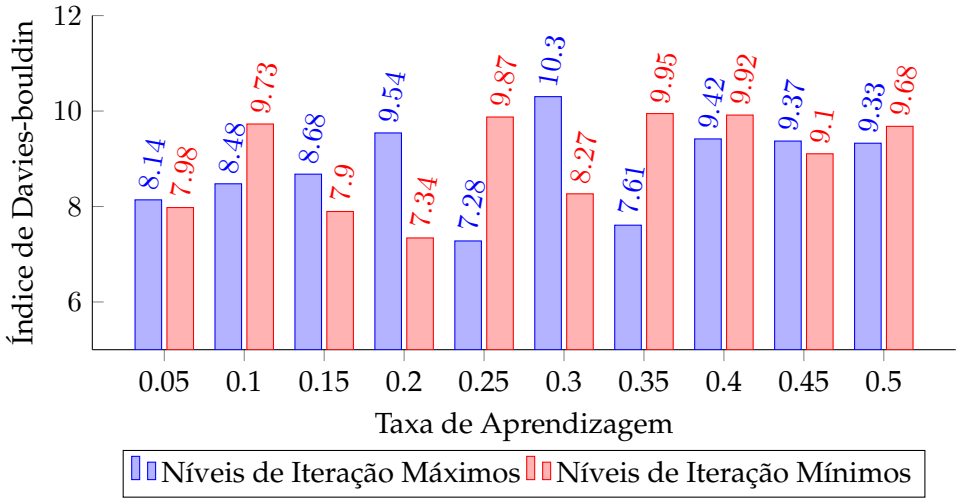
Figura 4.8: Histograma do Índice de Silhouette com variação do número de iterações usadas para treino do mapa auto-organizável.



4.6 Avaliação empírica por parte da AnubisNetworks

Relativamente ao trabalho efetuado e resultados apresentados, a resposta da AnubisNetworks foi positiva no sentido que é possível agrupar tráfego com padrões similares em *clusters* com eventos do mesmo género, sem haver intersecção de comunicação com os domínios de destino, foi mesmo possível observar, em capturas efetuadas a tráfego real, a formação de clusters que seguem um padrão semelhante de comunicação. Esta verificação foi feita por *experts* reconhecidos na área de segurança e de *threat intelligence*.

Figura 4.9: Histograma do Índice de Davies-bouldin com variação do número de iterações usadas para treino do mapa auto-organizável.



5

Conclusões

As *botnets* apresentam um conjunto de características juntamente com aplicação de métodos de evasão à deteção que dificultam a sua deteção. Nesta dissertação é apresentada uma relação entre o trabalho relacionado com soluções novas no contexto da deteção de *botnets*, permitindo assim desenhar novas formas de deteção. Foi apresentada uma solução que permite detetar *clusters* provenientes de *botnets* através do uso de classificadores de aprendizagem automática e correlação de informação proveniente de DNS e análises físicas de aplicações maliciosas.

A fonte de dados usada para a proposta do método e classificação baseia-se em tráfego de pedidos que deram resposta `NXDOMAIN` por parte do respetivo servidor DNS de operadores de telecomunicações. Este tipo de respostas é comum para aplicações maliciosas que recorrem a DGAs, no entanto este tipo de respostas geralmente é usado como mecanismo de recuperação em caso de falha do método de comunicação principal. Ao usar um algoritmo de *clustering* não supervisionado não é feita nenhuma assunção relativa aos dados, desta forma é possível relacionar tráfego de aplicações maliciosas analisadas no Maltracker com acesso a domínios não necessariamente gerados por DGAs. O classificador de DGAs devolve um mecanismo prático de classificação, facilitando o trabalho ao analista ao afirmar com 77,9% de certeza se um nome de domínio foi gerado por uma DGA ou não.

Os métodos descritos fazem parte do CONDENSER, esta metodologia reduz a dimensionalidade dos resultados obtidos e resulta num grafo para posterior análise por parte de um analista. Para esta metodologia é adicionalmente sugerido o uso de uma solução que guarda histórico de resoluções DNS de forma activa, a fim de obter tanto os domínios de uma *botnet* como os domínios e IPs ativos dos próprios C&Cs.

A classificação de nomes de domínio para cada *cluster* é feita à *priori* do algoritmo de

clustering, desta forma não é considerado o erro de classificação associado ao classificador de nomes de domínio, o que poderia influenciar a formação dos *clusters*.

5.1 Limitações

Devido à natureza dos dados usados não foi possível fazer uma avaliação comparativa tendo em conta o trabalho relacionado e a solução no seu todo. No entanto são apresentadas medidas de performance para cada um dos classificadores e para o algoritmo de *clustering* usado. Relativamente ao trabalho proposto por [DK13], não foi feita uma comparação em termos de performance dado que a lista proveniente do *web-site* `www.malwaredomains.com` é dinâmica, desta forma não é assegurada uma base de comparação justa em relação ao trabalho de Davut, et al.

Apesar de não ser um dos objetivos desta tese, foi observado que durante a implementação da metodologia proposta a escalabilidade é um factor predominante no sucesso da deteção devido a mecanismos de *fast-flux* e *double fast-flux*. A resolução de domínios de forma rápida é um requisito para uma deteção rápida de alterações nos registos presentes nos servidores DNS, razão pela qual a performance da deteção pode ser influenciada por incapacidade de processamento ou por atraso na resolução de domínios. Por esta razão tanto o CONDENSER como o DNS Crawler foram desenhados com escalabilidade em mente, de forma minimizar este tipo de problemas.

Tanto os fatores de escalabilidade como o problema de obtenção dos dados também é referido por Sommer, et al. [SP10], que argumenta que as soluções existentes na academia são muitas vezes baseados em estudo que recaem numa pequena porção de tráfego, sendo difícil transpor as soluções apresentadas para um ambiente de produção real.

Associado ao treino dos classificadores e do algoritmo de *clustering* usado também se encontram limitações no tempo de processamento, foi observado que os processos em questão estão limitados pela quantidade e núcleos de processamento disponíveis, pelo que para uma solução capaz de processar tráfego em larga escala, para o qual é necessário um treino periódico do *clusterer*, é recomendado que a solução recaia numa máquina com grande poder de processamento, em conjunto com uma solução de *clustering* paralelizada (*multi-threaded*).

Devido à natureza dos dados presentes nas diferentes *feeds* da AnubisNetworks (como a *feed* de `NXDOMAINS` e a de *sinkholing*) não é possível intersectar a sua informação, facto que impossibilitou a criação de um conjunto de dados tanto de tráfego de aplicações maliciosa, como de tráfego considerado legítimo para criação de um classificador.

5.2 Contribuições

Nesta dissertação são apresentados 11 atributos para classificação de nomes de domínio gerados por DGAs, que mostraram 77,9% de Precisão e 83,2% no cálculo da *F-Measure*. Neste trabalho são apresentados 81 atributos para *clustering*, a fim de detetar domínios

gerados por DGAs e de agrupar comunicações efetuadas por aplicações maliciosas. Foi também desenvolvida uma metodologia que pretende relacionar informações provenientes de análises físicas e de informação histórica de registos DNS, desta forma pretende-se correlacionar amostras reais de aplicações maliciosas com as informações recolhidas. Esta metodologia no seu todo oferece uma solução de inteligência que agrega informação que permite lidar com rotação de DGAs, aplicações maliciosas de última hora, conjunto de servidores usados como C&C para uma determinada *botnet* e deteção de *clusters* com domínios gerados por DGAs, sem haver necessidade de correlação com informação de DNS e de análises físicas.

5.3 Botconf

Foi aceite um *draft* inicial do trabalho descrito nesta dissertação na Botconf (`https://www.botconf.eu`). A Botconf é uma conferência da indústria que incide sobre o trabalho corrente e investigação sobre a área das *botnets*. Já vai na sua segunda edição e irá decorrer em Nancy, França. Esta conferência irá contar com o apoio da Agência Europeia de Segurança, a Europol, assim como inúmeras empresas do ramo da ciber-segurança. O artigo proposto irá incidir sobre uma interface web que permitirá visualizar o resultado do processo de *clustering* efetuado por intermédio de uma representação dos *clusters* em grafos, obtidos através dos métodos propostos nesta dissertação.

5.4 Trabalho Futuro

No futuro é aconselhado o uso de algoritmos de aprendizagem automática que não foram usados durante a elaboração deste estudo. Os GHSOM, A-GHSOM e GHSOM-1 são algoritmos de *clustering* hierárquicos com treino contínuo, que podem melhorar a classificação e treino dado que dispõem de mais camadas de representação dos dados e também porque se adaptam a variações observadas no conjunto de dados usado para treino. Também é sugerido o uso de uma alternativa que recorra a uma implementação dos algoritmos de aprendizagem automática em GPUs.

Adicionalmente é proposto o estudo do uso do algoritmo de *clustering k-Means* com aumento iterativo no número de clusters em que é feita uma monitorização progressiva dos valores obtidos nos índices de Dunn, Bouldin-davies e Silhouette. Assim é possível obter um conjunto de *clusters* considerado ótimo. No entanto este método tem a desvantagem de poder ser moroso em termos de cálculo, podendo no entanto haver alternativas em trabalho relacionado existente que tenham este facto em consideração. Relativamente ao trabalho de Stalmans, et al. [SI] é proposto um conjunto de atributos alternativo adaptado ao tipo de tráfego disponível, por se tratar de tráfego de domínios não existentes propõem-se o uso georeferenciado de coordenadas dos ccTLDs, atribuindo a cada TLD uma posição geográfica (e.g. *.kz* é o ccTLD do Cazaquistão, que teria a sua capital como coordenada geográfica).

É sugerido investigação adicional sobre atributos adicionais sobre os clusters formados no CONDENSER. Na Listagem 3.1 são apresentadas algumas métricas que podem auxiliar na criação de um classificador de *clusters*.

Toda a investigação apresentada nesta dissertação irá ser continuada no seio da AnubisNetworks, onde se pretende implementar uma solução semelhante ao sistema proposto para ser usado em ambiente de produção.

Bibliografia

[AJS06] M. Amini, R. Jalili e H. R. Shahriari. "RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks". Em: *Computers & Security* 25.6 (set. de 2006), pp. 459–468. ISSN: 01674048. DOI: 10.1016/j.cose.2006.05.003. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167404806000782.

[AH13] P. Amoli e T Hamalainen. "A real time unsupervised NIDS for detecting unknown and encrypted network attacks in high speed network". Em: *Measurements and Networking ...* (2013). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6663794.

[ADEJ12] M. Antonakakis, J. Demar, C. Elisan e J. Jerrim. "DGAs and Cyber-Criminals: A Case Study". Em: (2012), pp. 1–9. URL: http://www.wikitoo.com/downloads/r_pubs/RN_DGAs-and-Cyber-Criminals-A-Case-Study.pdf.

[AP12] M. Antonakakis e R. Perdisci. "From throw-away traffic to bots: detecting the rise of DGA-based malware". Em: *Proceedings of the 21st ...* (2012). URL: https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final127.pdf.

[AMLJS09] B. AsSadhan, J. M. Moura, D. Lapsley, C. Jones e W. T. Strayer. "Detecting Botnets Using Command and Control Traffic". Em: *2009 Eighth IEEE International Symposium on Network Computing and Applications* 4 (jul. de 2009), pp. 156–162. DOI: 10.1109/NCA.2009.56. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5190367.

[AB11] M. Augustin e A. Balaz. "Intrusion detection with early recognition of encrypted application". English. Em: *2011 15th IEEE International Conference on Intelligent Engineering Systems*. IEEE, jun. de 2011, pp. 245–247. ISBN: 978-1-4244-8954-1. DOI: 10.1109/INES.2011.

5954752. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5954752http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5954752.

[Bel] Belgacom Group. *Belgacom takes actions related to IT security - Belgacom Group*. URL: http://www.belgacom.com/be-en/newsdetail/ND_20130916_Belgacom.page.

[BBR12] L. Bilge, D. Balzarotti e W. Robertson. "Disclosure: detecting botnet command and control servers through large-scale NetFlow analysis". Em: *Proceedings of the 28th ...* (2012), pp. 129–138. URL: http://dl.acm.org/citation.cfm?id=2420969.

[CV95] C Cortes e V Vapnik. "Support-vector networks". Em: *Machine learning* 297 (1995), pp. 273–297. URL: http://link.springer.com/article/10.1007/BF00994018.

[Dat12] Data Genetics. *Domain Name Analysis*. 2012. URL: http://datagenetics.com/blog/march22012/.

[DB79] D. L. Davies e D. W. Bouldin. "A Cluster Separation Measure". Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (abr. de 1979), pp. 224–227. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1979.4766909. URL: http://dl.acm.org/citation.cfm?id=2053034.2053416http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4766909.

[DK13] N. Davuth e S. Kim. "Classification of Malicious Domain Names using Support Vector Machine and Bi-gram Method". Em: *sersc.org* 7.1 (2013), pp. 51–58. URL: http://www.sersc.org/journals/IJSIA/vol7_no1_2013/5.pdf.

[Des14] L. Deshotels. "Inaudible Sound as a Covert Channel in Mobile Devices 2014". Em: *8th USENIX Workshop on Offensive Technologies (WOOT 14)*. San Diego, CA: USENIX Association, 2014. URL: https://www.usenix.org/conference/woot14/workshop-program/presentation/deshotels.

[DRP12] C. Dietrich, C. Rossow e N. Pohlmann. "CoCoSpot: Clustering and recognizing botnet command and control channels using traffic analysis". Em: *Computer Networks* (2012), pp. 1–20. URL: http://www.sciencedirect.com/science/article/pii/S1389128612002472.

[DK32] H. Driver e A. Kroeber. "Quantitative Expression of Cultural Relationships". Em: *University of California Publications in American Archaeology and Ethnology* 31.4 (1932), pp. 211–256. URL: http://digitalassets.lib.berkeley.edu/anthpubs/ucb/text/ucp031-005.pdf.

[Dun74] J. C. Dunn†. "Well-Separated Clusters and Optimal Fuzzy Partitions". Em: *Journal of Cybernetics* 4.1 (jan. de 1974), pp. 95–104. ISSN: 0022-0280. DOI: 10.1080/01969727408546059. URL: http://dx.doi.org/10.1080/01969727408546059http://www.tandfonline.com/doi/abs/10.1080/01969727408546059.

[Fau94] L. Fausett. *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall, Inc., jul. de 1994, p. 461. ISBN: 0-13-334186-0. URL: http://dl.acm.org/citation.cfm?id=197023http://books.google.fr/books?id=ONylQgAACAAJ.

[FMHHB] T. Frosch, K Marc, T. Holz, G Horst e R.-u. Bochum. "Predentifier : Detecting Botnet C & C Domains From Passive DNS Data Motivation : DNS Features of Botnet Domains". Em: (), pp. 1–14.

[GPZL08] G. Gu, R. Perdisci, J. Zhang e W. Lee. "BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection". Em: *17th USENIX Security Symposium*. USENIX Association, jul. de 2008, pp. 139–154. DOI: 10.1.1.144.5167. URL: http://dl.acm.org/citation.cfm?id=1496711.1496721.

[HG13] M. Hanspach e M. Goetz. "On Covert Acoustical Mesh Networks in Air". Em: *Journal of Communications Vol. 8, No. 11, November 2013* 8.11 (2013). DOI: 10.12720/jcm.8.11.758-767.

[Hon12] L. V. Hong. "DNS Traffic Analysis for Network-based Malware Detection". Tese de doutoramento. Technical University of Denmark, 2012. URL: http://www.diva-portal.org/smash/record.jsf?pid=diva2:524298.

[IZ12] D. Ippoliti e X. Zhou. "A-GHSOM: An adaptive growing hierarchical self organizing map for network anomaly detection". Em: *Journal of Parallel and Distributed Computing* 72.12 (dez. de 2012), pp. 1576–1590. ISSN: 07437315. DOI: 10.1016/j.jpdc.2012.09.004. URL: http://www.sciencedirect.com/science/article/pii/S0743731512002195.

[KR10a] R. Koch e G. D. Rodosek. "Command Evaluation in Encrypted Remote Sessions". English. Em: *2010 Fourth International Conference on Network and System Security*. IEEE, set. de 2010, pp. 299–305. ISBN: 978-1-4244-8484-3. DOI: 10.1109/NSS.2010.62. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5635554http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5635554.

[KR10b] R. Koch e G. D. Rodosek. "Security system for encrypted environments (S2E2)". Em: *RAID'10 Proceedings of the 13th international conference on Recent advances in intrusion detection*. Vol. Recent Adv. Springer-Verlag, set. de 2010, pp. 505–507. ISBN: 3-642-15511-1, 978-3-642-15511-6. DOI: 10.1007/978-3-642-15512-3_35. URL: http://dl.acm.org/citation.cfm?id=1894166.1894209.

[Koh82] T. Kohonen. "Analysis of a simple self-organizing process". Em: *Biological Cybernetics* 44.2 (jul. de 1982), pp. 135–140. ISSN: 0340-1200. DOI: 10.1007/BF00317973. URL: http://link.springer.com/10.1007/BF00317973.

[Koh98] T. Kohonen. "The self-organizing map". Em: *Neurocomputing* 21.1-3 (nov. de 1998), pp. 1–6. ISSN: 09252312. DOI: 10.1016/S0925-2312(98)00030-7. URL: http://www.sciencedirect.com/science/article/pii/S0925231298000307.

[L. 90] P. J. R. L. Kaufman. *Finding Groups in Data: An Introduction to Cluster Analysis*. 1990. ISBN: 0-471-87876-6.

[LT12] K. Limthong e T. Tawsook. "Network traffic anomaly detection using machine learning approaches". Em: *Network Operations and ...* (2012), pp. 1–4. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6211951.

[LLS] C. Livadas, D. Lapsley e W. T. Strayer. "Using Machine Learning Techniques to Identify Botnet Traffic". Em: ().

[LM] J. Livingood e N. Mody. *Recommendations for the Remediation of Bots in ISP Networks*. Rel. téc. URL: http://tools.ietf.org/html/rfc6561.

[Mit97] T. M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997, p. 432. ISBN: 0070428077. URL: http://www.amazon.com/Machine-Learning-Tom-M-Mitchell/dp/0070428077.

[Moc83a] P. Mockapetris. "RFC 882 - Domain Names: Concepts and Facilities". Em: *Network Working Group* (1983), p. 31. URL: https://tools.ietf.org/html/rfc882.txt.

[Moc83b] P. Mockapetris. "RFC 883 - Domain names: Implementation specification". Em: *Network Working Group* (1983), p. 73. URL: http://tools.ietf.org/html/rfc883.

[Moc87a] P. Mockapetris. "RFC 1034 - Domain names: Concepts and Facilities". Em: *Network Working Group* (1987), p. 55. URL: http://tools.ietf.org/html/rfc1034.

[Moc87b] P. Mockapetris. "RFC 1035 - Domain names: Implementation and Specification". Em: *Network Working Group* (1987), p. 55. URL: http://tools.ietf.org/html/rfc1035.

[Moz] Mozilla Foundation. *Public Suffix List - MozillaWiki*. URL: https://wiki.mozilla.org/Public_Suffix_List.

[NPA13] T. Nelms, R. Perdisci e M. Ahamad. "ExecScent : Mining for New C & C Domains in Live Networks with Adaptive Control Protocol Templates with Adaptive Control Protocol Templates". Em: *22nd USENIX Security Symposium*. 2013, pp. 589–604. ISBN: 9781931971034.

[Nora] Normanshark. *Necurs - C&C domains non-censorable - Norman Shark*. URL: http://normanshark.com/blog/necurs-cc-domains-non-censorable/.

[Norb] Normanshark. *Necurs C&C - part-2 - Norman Shark*. URL: http://normanshark.com/blog/necurs-cc-part-2-2/.

[PLLDL10] E. Palomo, J. Ortiz-de Lazcano-Lobato, E. Dominguez e R. Luque. "An anomaly detection system using a GHSOM-1". Em: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul. de 2010, pp. 1–7. ISBN: 978-1-4244-6916-1. DOI: 10.1109/IJCNN.2010.5596967. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5596967http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5596967.

[RMD02] A Rauber, D Merkl e M Dittenbach. "The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data." English. Em: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 13.6 (jan. de 2002), pp. 1331–41. ISSN: 1045-9227. DOI: 10.1109/TNN.2002.804221. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=1058070http://www.ncbi.nlm.nih.gov/pubmed/18244531.

[RA11] E. Rendón e I. Abundez. "Internal versus External cluster validation indexes". Em: *International Journal of ...* 5.1 (2011). URL: http://w.naun.org/multimedia/UPress/cc/20-463.pdf.

[RU13] S. Ruehrup e P. Urbano. "Botnet detection revisited: theory and practice of finding malicious P2P networks via Internet connection graphs". Em: *...IEEE Conference on* Tma (2013), pp. 435–440. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6562902.

[SRS07] S. Saitta, B. Raphael e I. F. Smith. *Machine Learning and Data Mining in Pattern Recognition*. Ed. por P. Perner. Vol. 4571. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, jul. de 2007, pp. 174–187. ISBN: 978-3-540-73498-7. DOI: 10.1007/978-3-540-73499-4. URL: http://dl.acm.org/citation.cfm?id=1420326.1420344.

[SM07] K. Scarfone e P. Mell. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. Rel. téc. National Institute of Standards e Technology, 2007, p. 127. URL: http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf.

[SMCZ13] S. Schiavoni, F Maggi, L Cavallaro e S. Zanero. "Tracking and Characterizing Botnets Using Automatically Generated Domains". Em: *arXiv preprint arXiv: ...* (2013). arXiv: arXiv:1311.5612v1. URL: http://arxiv.org/abs/1311.5612.

[SSPS13] S. S. Silva, R. M. Silva, R. C. Pinto e R. M. Salles. "Botnets: A survey". Em: *Computer Networks* 57.2 (fev. de 2013), pp. 378–403. ISSN: 13891286. DOI: 10.1016/j.comnet.2012.07.021. URL: http://linkinghub.elsevier.com/retrieve/pii/S1389128612003568.

[Sma] Smart Insights. *Mobile marketing statistics 2013*. URL: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/.

[SP10] R. Sommer e V. Paxson. "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection". English. Em: *2010 IEEE Symposium on Security and Privacy*. IEEE, 2010, pp. 305–316. ISBN: 978-1-4244-6894-2. DOI: 10.1109/SP.2010.25. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5504793.

[SZ10] S. Sparks e C. C. Zou. "An Advanced Hybrid Peer-to-Peer Botnet". English. Em: *IEEE Transactions on Dependable and Secure Computing* 7.2 (abr. de 2010), pp. 113–127. ISSN: 1545-5971. DOI: 10.1109/TDSC.2008.35. URL: http://www.computer.org/csdl/trans/tq/2010/02/ttq2010020113.htmlhttp://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4569852.

[SI] E. Stalmans e B. Irwin. "Spatial Statistics as a Metric for Detecting Botnet C2 Servers". Em: *botconf.eu* (). URL: https://www.botconf.eu/wp-content/uploads/2013/08/09-EtienneStalmans-paper.pdf.

[SGCCGSKKV09] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel e G. Vigna. "Your botnet is my botnet". Em: *Proceedings of the 16th ACM conference on Computer and communications security - CCS '09*. New York, New York, USA: ACM Press, nov. de 2009, p. 635. ISBN: 9781605588940. DOI: 10.1145/1653662.1653738. URL: http://dl.acm.org/citation.cfm?id=1653662.1653738.

[Sym14] Symantec. *Trojan.Cryptolocker | Symantec*. 2014. URL: http://www.symantec.com/security_response/writeup.jsp?docid=2013-091122-3112-99.

[Tru13] Trustwave. *2013 Global Security Report*. Rel. téc. 2013. URL: http://www2.trustwave.com/rs/trustwave/images/2013-Global-Security-Report.pdf.

[XBLXT11] C. Xiang, F. Binxing, Y. Lihua, L. Xiaoyi e Z. Tianning. "Andbot: towards advanced mobile botnets". Em: *LEET'11 Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats*. USENIX Association Berkeley, CA, USA ©2011, mar. de 2011, p. 11. URL: http://dl.acm.org/citation.cfm?id=1972441.1972456.

[ZTSLSGG13] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani e D. Garant. "Botnet detection based on traffic behavior analysis and flow intervals". Em: *Computers & Security* 39 (nov. de 2013), pp. 2–16. ISSN: 01674048. DOI: 10.1016/j.cose.2013.04.007. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167404813000837.

A

Performance dos Classificadores

#	Tamanho do Sub-Conjunto	*Accuracy*	Precisão	*Recall*	*F-Measure*
1	4	54,8489	54,6418	55,5969	55,1152
2	10	60,5229	59,7222	63,9643	61,7705
3	22	75,6536	72,4104	82,6701	77,2009
4	46	75,2271	71,6342	83,3023	77,0289
5	94	75,4497	70,6881	86,7237	77,8891
6	188	75,8391	71,1538	86,6865	78,1559
7	378	79,1953	71,9284	95,5746	82,0824
8	758	79,7515	73,0580	94,0870	82,2496
9	1518	79,8257	73,8884	92,0788	81,9867
10	3036	79,9555	73,0505	94,7564	82,4996
11	6072	80,6045	74,1689	93,7523	82,8187
12	12146	81,5872	77,1621	89,5872	82,9117
13	24294	81,7727	77,3047	89,8103	83,0896
14	48588	81,7912	77,6303	89,1781	83,0045
15	53934	82,0508	77,9474	89,2525	83,2178

Tabela A.1: Performance do Classificador do Nomes de Domínio com Partições de Dimensão Exponencial

#	Tamanho do Sub-Conjunto	*Accuracy*	Precisão	*Recall*	*F-Measure*
1	6478	80,7157	74,2714	93,8267	82,9116
2	12956	81,6985	77,2407	89,7360	83,0208
3	19434	81,7356	77,5907	89,1038	82,9497
4	25912	81,7727	77,1656	90,1078	83,1360
5	32390	81,7727	77,7489	88,8806	82,9429
6	38868	81,7727	77,5695	89,2525	83,0019
7	45346	81,7356	77,5016	89,2897	82,9791
8	51824	81,8468	77,5097	89,5872	83,1120
9	53934	82,0508	77,9474	89,2525	83,2178

Tabela A.2: Performance do Classificador do Nomes de Domínio com Partições de Dimensão Linear

#	Tamanho do Sub-Conjunto	*Accuracy*	Precisão	*Recall*	*F-Measure*
1	2	59,7455	55,7870	97,2755	70,9084
2	4	92,0102	94,7425	89,1019	91,8357
3	8	69,0585	63,3079	91,9273	74,9794
4	16	92,5191	92,7992	92,3310	92,5645
5	34	89,3130	91,9441	86,3774	89,0739
6	68	90,2799	92,3839	86,8819	89,5485
7	138	91,5522	93,9297	89,0010	91,3990
8	276	90,6361	94,7836	86,1756	90,2749
9	552	95,9796	96,4444	97,7800	97,1076
10	1106	95,9796	95,5090	96,5691	96,0361
11	2212	96,4885	95,8250	97,2755	96,5448
12	4426	96,4377	95,7299	97,2755	96,4965
13	8852	96,6412	96,1117	97,2755	96,6901
14	17704	96,6921	96,2076	97,2755	96,7386
15	19658	96,7430	96,3037	97,2755	96,7872

Tabela A.3: Performance do Classificador de URIs com Partições de Dimensão Exponencial

#	Tamanho do Sub-Conjunto	*Accuracy*	Precisão	*Recall*	*F-Measure*
1	2360	96,0814	95,5179	96,7709	96,1403
2	4720	96,4377	95,7299	97,2755	96,4965
3	7080	96,6412	96,1117	97,2755	96,6901
4	9440	96,6412	96,1117	97,2755	96,6901
5	11800	96,6412	96,1117	97,2755	96,6901
6	14160	96,6921	96,2076	97,2755	96,7386
7	16520	96,6921	96,2076	97,2755	96,7386
8	18880	96,7430	96,2076	97,2755	96,7386
9	19658	96,7430	96,2076	97,2755	96,7386

Tabela A.4: Performance do Classificador de URIs com Partições de Dimensão Dimensão

B

Atributos Seleccionados para Extracção

Tabela B.1: Análise do Conjunto de Dados Pertencentes à Classe de Domínios Normais

Atributo	μ	$\overline{x}$	σ	$CI_{0.95}(1-\alpha)$
`consonantRatio`	0.2289	0.2289	0.2709	[0.2256 ; 0.2321]
`consonantVocalRatio`	0.2627	0.2628	0.2698	[0.2595 ; 0.2660]
`domainLength`	-0.9317	-0.9318	0.0294	[-0.9322 ; -0.9314]
`othersRatio`	-0.9883	-0.9883	0.0508	[-0.9890 ; -0.9877]
`vocalRatio`	-0.2788	-0.2788	0.2653	[-0.2820 ; -0.2756]
`digitRatio`	-0.9616	-0.9616	0.1679	[-0.9636 ; -0.9596]
`noRepeatsByUniGram`	-0.9454	-0.9455	0.0194	[-0.9457 ; -0.9453]
`noRepeatsByUniGram`	-0.9405	-0.9405	0.0286	[-0.9408 ; -0.9402]
`noRepeatsByUniGram`	-0.9473	-0.9473	0.0295	[-0.9477 ; -0.9470]
`tetraGramAnalysis`	-0.8771	-0.8771	0.3440	[-0.8812 ; -0.8729]
`lowFrequenceOccurrence`	-0.9981	-0.9981	0.0611	[-0.9988 ; -0.9973]

Tabela B.2: Análise do Conjunto de Dados não Pertencentes à Classe de Domínios Normais

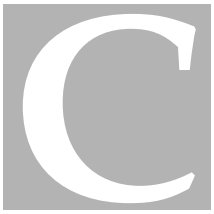
Atributo	μ	$\overline{x}$	σ	$CI_{0.95}(1-\alpha)$
consonantRatio	0.4357	0.4357	0.3297	[0.4317 ; 0.4396]
consonantVocalRatio	0.5120	0.5120	0.2757	[0.5087 ; 0.5152]
domainLength	-0.8738	-0.8737	0.0781	[-0.8747 ; -0.8728]
othersRatio	-0.9972	-0.9972	0.0206	[-0.9975 ; -0.9970]
vocalRatio	-0.5350	-0.5350	0.2787	[-0.5383 ; -0.5317]
digitRatio	-0.9033	-0.9033	0.2660	[-0.9065 ; -0.9001]
noRepeatsByUniGram	-0.9100	-0.9100	0.0429	[-0.9105 ; -0.9095]
noRepeatsByUniGram	-0.8836	-0.8836	0.0756	[-0.8845 ; -0.8827]
noRepeatsByUniGram	-0.8890	-0.8889	0.0782	[-0.8899 ; -0.8880]
tetraGramAnalysis	-0.4242	-0.4242	0.5931	[-0.4313 ; -0.4171]
lowFrequenceOccurrence	-0.9883	-0.9883	0.1524	[-0.9901 ; -0.9864]

Tabela B.3: Análise do Conjunto de Dados Pertencentes à Classe de URIs Normais

Atributo	μ	$\overline{x}$	σ	$CI_{0.95}(1-\alpha)$
queryLength	-0.9913	-0.9913	0.0515	[-0.9923 ; -0.9903]
queryArgumentSize	-0.9778	-0.9778	0.0758	[-0.9793 ; -0.9763]
uriBaseLength	-0.9389	-0.9389	0.0595	[-0.9401 ; -0.9377]
uriBaseLevelLength	-0.6693	-0.6693	0.1635	[-0.6725 ; -0.6661]
uriPathLength	-0.9316	-0.9316	0.0715	[-0.9330 ; -0.9302]

Tabela B.4: Análise do Conjunto de Dados não Pertencentes à Classe de URIs Normais

Atributo	μ	$\overline{x}$	σ	$CI_{0.95}(1-\alpha)$
queryLength	-0.9464	-0.9464	0.2067	[-0.9505 ; -0.9423]
queryArgumentSize	-0.8822	-0.8822	0.2924	[-0.8880 ; -0.8764]
uriBaseLength	-0.9660	-0.9660	0.1204	[-0.9684 ; -0.9636]
uriBaseLevelLength	-0.8852	-0.8852	0.0936	[-0.8870 ; -0.8833]
uriPathLength	-0.9381	-0.9381	0.1616	[-0.9413 ; -0.9349]



Atributos Relevantes no Trabalho Relacionado

Tabela C.1: Análise aos Nomes de Domínio

Descrição	Autores
Bi-Gram Score - Pontuação à ocorrência de repetição de caracteres, foi definida uma margem para reduzir ruído à classificação.	[DK13]
Domain Reputation - Sistema de Reputação de domínios, através do sistema EXPOSURE.	[SMCZ13]
Prefix Chars Ratio - Rácio de Caracteres Significantes com um determinado prefixo.	[SMCZ13]
n-Gram Score - Pontuação por atribuição de um número de n-gramas observados.	[SMCZ13]
Mahalanobis Distance - Medida de Distância de atributos individuais pela distância de Mahalanobis. Também foi associado uma margem de valores para uma classificação mais precisa.	[SMCZ13]
IP Address - Endereço IP do C&C para DGA *Fingerprinting*.	[SMCZ13]
Domain Prefix Length - Comprimento do prefixo do domínio para DGA *Fingerprinting*.	[SMCZ13]
Digit Ratio - Número de dígitos comparado com o comprimento do domínio.	[FMHHB]
Consonant Ratio - Número de consoantes comparado com o comprimento do domínio.	[FMHHB]
Consonant Vocal Ratio - Número de consoantes comparado com o número de vogais.	[FMHHB]

Tabela C.2: Análise ao Conteúdo de Pacotes

Tipo	Descrição	Autor
Pacote IP	`diff-tme-stamp`, `ip-id`, `ip-tos`, `ip-ttl`, `ip-headerlen`, `ip-len`, `is-home-source-ip`, `is-home-dest-ip`, `is-land`, `ip-frag-flag`	[AJS06]
Pacote TCP	`tcp-src-port`, `tcp-dest-port`, `tcp-fin`, `tcp-syn`, `tcp-rst`, `tcp-push`, `tcp-ack`, `tcp-urg`, `tcp-offset`, `tcp-win-size`	[AJS06]
Pacote UDP	`udp-src-port`, `udp-dest-port`	[AJS06]
Pacote ICMP	`icmp-type`, `icmp-code`, `icmp-id`, `icmp-sequence`	[AJS06]
Packet Length	Comprimento dos quatro primeiros pedidos HTTP para cada fluxo.	[DRP12]
Distinct Bytes	Número de bytes distintos no parâmetro *query* de um URI.	[DRP12]
Response Body Length	Comprimento do corpo da resposta.	[DRP12]

Tabela C.3: Análise a Informação do Sistema de Nomes de Domínio

Descrição	Autores
IP Address - Endereço IP do C&C para DGA *Fingerprinting*.	[SMCZ13]
IPCount - Número de IPs máximo que o domínio resolve.	[FMHHB]
T/L Max - Tempo de vida máximo durante a observação (em seg).	[FMHHB]
T/L Min - Tempo de vida mínimo durante a observação (em seg).	[FMHHB]
T/L Diff - Diferença do tempo de vida máximo com o mínimo (em seg).	[FMHHB]
SOA S/N Changes - Number of SOA S/N changes during observation.	[FMHHB]
Exp Time Min - Tempo de validade mínimo de uma zona do domínio (em seg).	[FMHHB]
Refresh Time Min - Tempo mínimo de actualização da zona do domínio (em seg).	[FMHHB]
Country Count - Número de países de onde o domínio é servido.	[FMHHB]
ASN Count - Número de ASNs de onde o domínio é servido.	[FMHHB]
Domain Age - Tempo de vida do domínio em dias (ultima vez visto - data de criação).	[FMHHB]
Latitude - Latitude do IP de um registo de tipo A, obtido através de GeoIP.	[SI]
Longitude - Longitude do IP de um registo de tipo A, obtido através de GeoIP.	[SI]
Posição UMT - Posição Universal Transversa de Mercator de um IP, obtido através de GeoIP.	[SI]
Localização MGRS - Sistema de coordenadas geográficas usada pela NATO.	[SI]