Métodos de Detecção de Malware por Análise de Tráfego

Luís Carlos Marques Afonso

Relatório Final do Trabalho de Projeto apresentado à

Escola Superior de Tecnologia e de Gestão

Instituto Politécnico de Bragança

para obtenção do grau de Mestre em

Sistemas de Informação

Janeiro 2015

Métodos de Detecção de Malware por Análise de Tráfego

Luís Carlos Marques Afonso

Relatório Final do Trabalho de Projeto apresentado à

Escola Superior de Tecnologia e de Gestão

Instituto Politécnico de Bragança

para obtenção do grau de Mestre em

Sistemas de Informação

Orientadores:

Rui Pedro Lopes

Tiago Pedrosa

Janeiro 2015

Resumo

Em 1983, Len Eidelmen apresentou num seminário de segurança computacional, o primeiro programa informático com capacidade de se auto-replicar, conseguindo instalar-se em vários locais do sistema. Um ano depois, o termo vírus de computador foi definido como um programa que infecta outros programas, modificando-os para que seja possível instalar cópias de si mesmo.

Atualmente conhecidos como *malware*, termo utilizado para definir uma classe de programas informáticos criados com o intuito de causar danos ou até mesmo roubo de informação, são nos dias de hoje, uma das ameaças constantes nas novas tecnologias, sejam elas redes informáticas de grandes e pequenos portes, computadores pessoais e interfaces móveis.

Fato que torna crucial a compreensão desse esse tipo de software para que possam ser criadas medidas de segurança cada vez mais eficazes e que ajudem aos administradores de sistemas informáticos a agirem antes que existam prejuízos de grandes proporções.

A deteção do *malware* é um desafio, existem muitos programas que possuem um comportamento que se assemelha ao *malware* sem o serem, além disso constantes mutações no código do *malware* torna esse trabalho muito mais penoso.

Este trabalho propõe o estudo das metodologias aplicadas pelos diversos investigadores que a partir da analise de tráfego capturado em uma rede de computadores, ajudem a reconhecer padrões que permitam criar ou modificar regras de deteção de *malware*.

Abstract

In 1983, Len Eidelmen presented at a seminar of computer security the first computer program with the ability to self-replicate, managing to settle in various parts of the system. A year later the term computer virus was defined as a program that infects other programs by modifying them so that you can install copies of itself.

Currently known as malware, a term used to define a class of computer programs created with the intent to cause damage or even theft of information are today, one of the constant threats on new technology, whether computer networks of large and small sizes, personal computers and mobile interfaces.

Fact which makes it crucial to understand that this type of software to increasingly effective security measures can be created and that will help administrators of computer systems to act before there is any loss of large proportions.

The malware detection is a challenge, there are many programs that have a behavior that resembles malware without being also contained mutations in the malware code makes this job much more arduous.

This work proposes the study of the methodologies used by the various researchers from allowing the analysis of captured traffic on a network of computers to recognize patterns that allow us to create rules or modify existing rules in order to generate alerts when it finds some kind of abnormality usually caused by software considered suspicious or malignant.

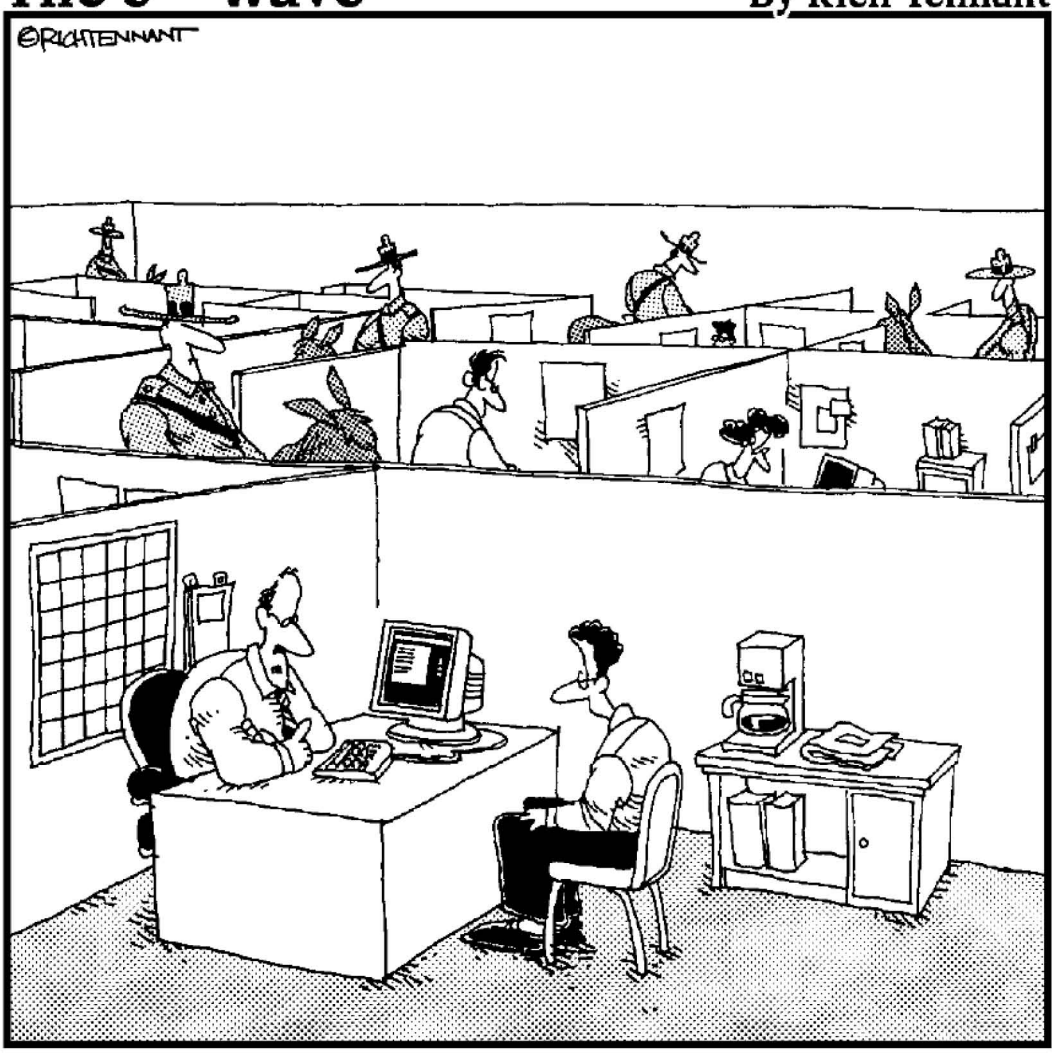
Aos meus pais em especial a minha querida mãe

minha irmãzinha e meu amado sobrinho,

que com seu apoio me fizeram superar as dificuldades.

The 5th Wave

By Rich Tennant



"We take network security very seriously here."

Agradecimentos

No início desta dissertação de mestrado, não poderia deixar de agradecer a todas as pessoas que, de alguma forma, colaboraram para a realização deste trabalho.

Primeiramente, dirijo aos meus orientadores, Professor Doutor Rui Pedro Lopes e Professor Doutor Tiago Pedrosa, um sincero agradecimento por acreditar e confiar, pelo apoio incansável, pela capacidade de ensinar a pensar, e pela forma amigável com que sempre se disponibilizaram para discutir os assuntos relacionados com esta dissertação.

À Universidade FUMEC, meus agradecimentos pela acolhida em suas instalações, em especial ao Professor Doutor Luiz Claudio Gomes Maia, meus agradecimentos pela acolhida no Mestrado e pela confiança depositada.

À minha família, em particular, à minha mãe e a minha irmã, pela paciência, compreensão, pelo incentivo e apoio incondicional que sempre me deram até atingir esta fase.

Aos meus amigos, o meu muito obrigado, pela força e pela ajuda que me forneceram, e sobretudo pela amizade.

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de Abreviaturas

AV	Antivirus.
CARO	Computer Antivirus Rearchers Organizatios.
CRM	Customer Relationship Management.
CSV	Comma-separated values.
DDoS	Distribution Deny of Service.
DNS	Domain Name System.
DOS	Denial of Setvice.
ERP	Enterprise Resource Planning.
HTTP	HyperText Transfer Protocol.
HTTPS	HyperText Transfer Protocol Secure.
ICMP	Internet Control Message Protocol.
IDS	Intrusion Detection System.
IP	Internet Protocol.
NADA	Network Anomaly Detection Algorithim.

OSI	Open Systems Interconnection.
PCAP	Packet CAPture.
RAM	Random Access Memory.
RAT	Remote Administration Tools.
RNA	Redes Neurais Artificiais.
SVM	Support Vector Machines.
TCP	Transmission Control Protocol.
ToS	Type of Service.
UDP	User Datagram Protocol.
VLAN	Virtual Local Area Network.

Capítulo 1

Introdução

Com o aumento significativo em tamanho e importância decorrido nos últimos anos na utilização de redes informáticas, tanto por parte de particulares como instituições, com a finalidade de ajudar no dia a dia, foi uma das grandes revoluções da sociedade. Contudo, essa constante migração de dados e serviços levou, naturalmente, ao aparecimento de software que procura, de alguma forma, tirar proveito de vulnerabilidades existentes nessas redes, com o propósito de roubar informação valiosa ou simplesmente de causar danos que na maioria das vezes provocam prejuízos significativos aos utilizadores lesados. Tipicamente, a informação mais procurada inclui obtenção de senhas de acesso, planos de negócios, dados relacionados com contas bancárias, etc (Holz et al., 2008). Este software malicioso é designado por *malware* (*malicious software*).

Com o surgimento desse software malicioso, a identificação, análise e tratamento desse tipo de software no menor espaço de tempo possível, tornou-se parte essencial do dia a dia dos administradores de redes, sendo uma das áreas de maior importância das instituições.

Pretende-se, com este trabalho, estudar os métodos de análise e classificação do *malware* existente e utilizar essas técnicas de análise, adaptando-as de forma a compreender o comportamento desse mesmo tráfego, com o intuito de detetar as anomalias que possam existir.

1.1 Motivação

De acordo com MyCERT (2013), o número de incidentes devidos à atuação de *malware* tem sido uma constante (Figura 1.1). Estudos realizados mostram que mais de 90% dos programas maliciosos não possuem descrição e a enorme maioria destes programas utiliza nomes parecidos com as aplicações e serviços tipicamente existentes nos sistemas operativos, com o intuito de iludir os utilizadores. Geralmente estão instalados em alguma pasta de sistema e, de forma autónoma, migram para outras pastas, procurando ficar sempre ativo independentemente de quem esteja a utilizar o equipamento.

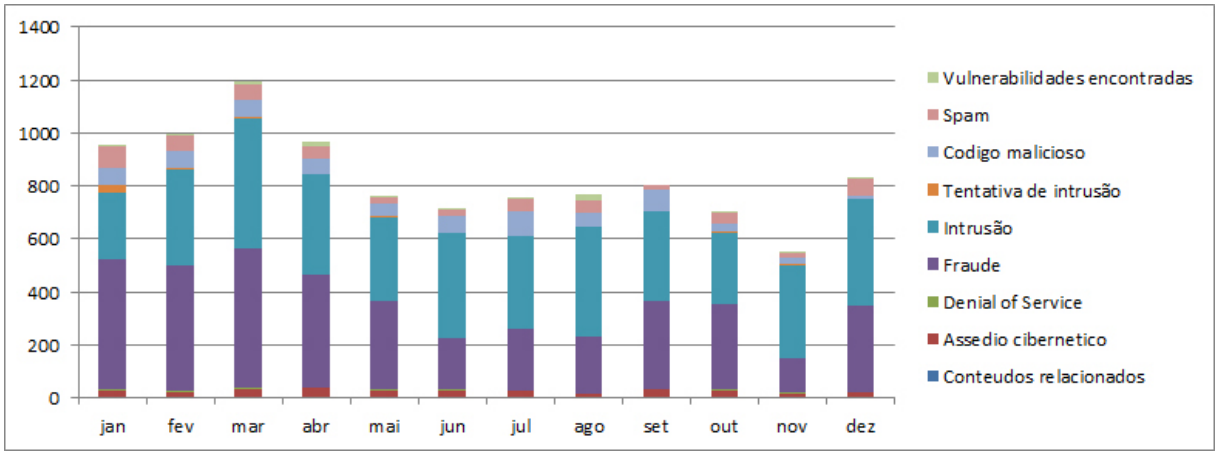


Figura 1.1: Gráfico dos incidentes reportados em 2012 - Fonte: MyCERT (2013)

Adicionalmente, o constante aumento do número e variabilidade do *malware* torna mais complicada a sua deteção e adaptação a novas estirpes.

1.2 Enquadramento

A constante migração de dados e serviços para a Internet levou, naturalmente, ao aparecimento de software que procura, de alguma forma, tirar proveito de vulnerabilidades existentes com o propósito de causar danos ou roubar informação.

Uma rede de comunicação consiste num conjunto de equipamentos ligados entre si,

fisicamente através de cabos ou outro sistema que permita a troca de informação entre os equipamentos ligados a essa rede. Essa troca de informação designada tráfego de rede, é realizada com a transmissão de uma sequência de pacotes Transmission Control Protocol (TCP), começando e terminando em algum tempo bem definido, no qual a informação viaja de um endereço origem para um endereço destino e vice-versa, único para cada equipamento em uma rede pública ou local (Internet Protocol (IP)). Esse tráfego pode ser considerado normal ou anómalo dependendo de certos parâmetros como por exemplo o numero elevado de bytes transmitidos, ou um pedido de acesso a uma máquina não autorizado, etc.

1.3 Objetivos

O objetivo é compreender de que forma o tráfego de rede pode ser analisado e quais as técnicas que podem ser utilizadas para detetar e classificar anomalias encontradas nesse tráfego. Com a informação resultante dessa análise, as organizações podem criar regras de segurança contra o *malware* existente, recorrendo à deteção e contenção de ameaças provocadas por estes, de uma forma transversal a todo o equipamento e sistemas. A abordagem base passa por analisar o tráfego de forma a verificar se alguma comunicação é gerada por equipamento infetado por *malware*. Assim, o estudo e avaliação de algumas técnicas utilizadas é fundamental, passando por um processo de análise de tráfego e aprendizagem de deteção de anomalias, utilizando ferramentas e algoritmos de programação utilizados pela comunidade cientifica. Existe porém uma necessidade forte de obter fidedignos para fazer a aprendizagem, considerando que existem muitos dados de teste já capturados durante eventos de segurança ou obtidos de forma dissimulada, o que leva com que esses dados sejam transversais. São redes privadas, não existindo acesso à internet, de difícil utilização em cenário real. Para ser possível experimentar algumas das técnicas encontradas na literatura consultada, foi necessário encontrar uma forma automatizada de capturar

dados reais para fazer essa aprendizagem, capturando pacotes reais em máquinas infetada por intermédio do método de assinaturas mais utilizado com a ferramenta *snort*, que nos permitiu classificar pacotes de tráfego com possível *malware*. Assim sendo, um dos objetivos deste trabalho passou por verificar se é possível, utilizando o conhecimento existente da deteção baseada em assinaturas, utilizar essas assinaturas para classificar o tráfego capturado, criar modelos de aprendizagem para classificar tráfego desconhecido. Para o efeito, utilizamos um laboratório virtual de estudo de tráfego de rede onde vamos capturar, analisar *malware* em uma rede virtual, averiguar os efeitos deste no tráfego nessa mesma rede, descobrir a sua assinatura ou determinar o seu comportamento com o intuito de criar modelos de segurança que possam ser utilizadas para a contenção do *malware*.

1.4 Estrutura do documento

Este documento está estruturado por capítulos sendo que este capítulo faz uma introdução ao tema deste trabalho onde são apresentadas algumas definições, as motivações, o problema e os objetivos para este trabalho.

No Capítulo 2 falaremos sobre *malware*, iremos apresentar modelos de classificação de software malicioso de acordo com características dos principais tipos existentes, mostrando uma taxonomia relevante para este trabalho.

No Capítulo 3 falaremos sobre métodos de classificação de tráfego de rede. Aqui serão abordados alguns métodos utilizados para a classificação de tráfego anómalo em circuito de rede.

O Capítulo 4 terá um teor mais prático, onde se irá utilizar o conhecimento adquirido neste trabalho para analisar e classificar tráfego de rede.

O Capítulo 5 apresenta nossa conclusão em relação aos métodos apresentados e ideias para trabalho futuro.

Capítulo 2

Malware

O propósito da análise de *malware* é obter informações essenciais para proteger uma rede informática de possíveis ataques aos sistemas. Seu objetivo é determinar com exatidão o que está acontecendo, alertando os responsáveis do sistema sobre possíveis anomalias no funcionamento normal da rede.

Este capítulo descreve os conceitos necessários e essenciais no desenvolvimento deste trabalho, fazendo uma classificação do *malware* existente para melhor compreensão dos diversos comportamentos.

Um artigo publicado em 1984 com o título de "Experiments with Computer Viruses"de Frederick B. Cohen, conta que o primeiro vírus informático foi desenvolvido para uma demonstração de **Leonard Adleman** (um dos desenvolvedores do algoritmo RSA[1]) em um seminário de segurança informática. Em 10 de novembro do mesmo ano, depois de uma semana de trabalho utilizando o sistema Unix em um VAX 11/750, o primeiro vírus é executado obtendo permissões sobre o sistema (Laboratório da ESET Latinoamérica, 2012).

Publicando seus estudos em 1986, Cohen define pela primeira vez os vírus, termo que

[1]**RSA** é um algoritmo de criptografia de dados, que deve o seu nome a três professores do Instituto MIT (fundadores da actual empresa RSA Data Security, Inc.), Ronald Rivest, Adi Shamir e Leonard Adleman, que inventaram este algoritmo

tem sido aplicado desde a publicação dos trabalhos de Adleman. A definição proposta por Cohen e aceite pela comunidade foi:

> *"Programa que pode infetar a outros programas incluindo uma cópia possivelmente evoluída de si mesmo."*

Hoje, com a expansão dos sistemas de comunicação e desenvolvimento de dispositivos móveis com características computacionais cada vez mais elevadas, juntamente com a facilidade de estar conectado à internet a qualquer hora e lugar, a intensificação das relações socioeconómicas em âmbito mundial, consequência desse avanço tecnológico, que possibilitou uma maior integração económica e cultural entre pessoas e organizações de diferentes pontos do planeta, fez com que surgissem pessoas com o intuito de causar dano ou tirar algum proveito da informação que utilizadores e organizações possuem armazenadas nos seus equipamentos, utilizando código malicioso.

Entende-se por código malicioso um conjunto de aplicações e outro tipo de programas que tem o intuito de comprometer a segurança de um sistema. A esse código ou programa dá-se a nome de *malware*, que deriva das palavras do inglês *malicious* e *software* e serve para designar qualquer software destinado a infiltrar-se num sistema computacional de forma ilícita, com o único intuito de causar dano ou roubar informações (Franklin et al., 2007).

Para facilitar a compreensão do comportamento deste tipo de código, é necessário classifica-lo para elaborar uma metodologia com políticas de segurança que ajudam na prevenção da ação indesejada desse código. O método mais comum utilizado na identificação de *malware* é o levantamento de suas ações num sistema comprometido. A sua análise pode ser dividida em dois tipos (Filho et al., 2010):

Análise estática: que obtém informações sobre o código malicioso sem que seja necessário executa-lo;

Análise dinâmica: onde o comportamento do código malicioso é monitorizado durante sua execução.

Em geral os códigos maliciosos podem ser classificados de acordo com alguma característica específica do seu comportamento (Melo et al., 2011). No entanto enquadrar um determinado exemplar de *malware* em uma única classe é um processo difícil de ser realizado, devido à evolução do código e à facilidade de se adicionar novas funcionalidades que dificultam esse processo. Além disso, a nova geração de *malware* possui a capacidade de desabilitar os programas de antivírus, ocultando-se no sistema o que dificulta ainda mais sua identificação.

A taxonomia apresentada neste documento é utilizada para se referir a certos tipos de *malware*, e identificadores atribuídos por mecanismos antivírus com base no comportamento dos exemplares ao longo do seu ciclo de vida.

O software malicioso pode ser dividido primordialmente através de sua dependência ou não do hospedeiro como descreve Melo et al.. Utilizam ferramentas de comunicação para se espalharem. Os *worms*, por exemplo, são enviados por correio eletrónico ou através de mensagens instantâneas, os *trojans* são provenientes de websites e os vírus são obtidos por download de ficheiros infetados. É bastante comum encontrar *malware* que possuem capacidades híbridas, com características de acordo com as suas necessidades. Algumas estirpes de *malware* tentam explorar as vulnerabilidades existentes nos sistemas, permitindo uma invasão não autorizada e ou mais privilégios ao sistema.

A intenção de qualquer *malware* é permanecer incógnito, ocultando-se ativamente ou simplesmente não se fazendo notar no sistema infetado.

Mais de 90% dos programas maliciosos que se podem encontrar não possui descrição. A maioria destes programas utiliza nomes parecidos às aplicações e serviços nos sistemas operativos, com o intuito de iludir os utilizadores. Estão instalados geralmente em alguma pasta de sistema e se auto copiam para outras pastas. Isso garante que o *malware* fique sempre ativo, independentemente de quem estiver utilizando o equipamento.

Malware como já foi dito anteriormente, trata-se de um código de software que tem como finalidade explorar alguma vulnerabilidade, permitindo ao seu criador ter acesso a informação contida no equipamento invadido. Este tipo de software pode causar efeitos que vão desde um aumento dos processos executados no equipamento, causando uma diminuição considerável na performance do equipamento atacado, até ao aumento significativo na transmissão de informação. Isso é considerado como uma anomalia, que em redes informáticas se referem tipicamente, a circunstâncias onde as operações da rede se distanciam do padrão considerado normal (Souza, 2008).

Detecção de anomalias é definida como o processo de análise de dados com um comportamento considerado diferente do comportamento normal esperado num determinado conjunto ao qual pertence. Seguindo esse raciocínio, a detecção de anomalias é definida como a tarefa de determinar discrepâncias entre o comportamento encontrado e o comportamento esperado na rede (Souza, 2008).

Intrusão é definida como o conjunto de ações desencadeadas pelo atacante, comprometendo a estrutura básica da segurança da informação, como por exemplo a integridade, confidencialidade e disponibilidade de determinada informação de um sistema informático. O atacante é definido como um qualquer utilizador ou entidade que invade um sistema sem autorização, com o intuito de provocar atos ilícitos. Detetar um atacante ou intruso é uma tarefa complexa, uma vez que este pode possuir em seu poder a identificação de um utilizador legitimo, ou se comportar como tal, podendo ser classificados como internos[2] ou externos[3] (Silva et al., 2001).

Detecção de intrusão é o processo de detectar o uso não autorizado ou ataques em um sistema ou rede informática. Um sistema de detecção ou Intrusion Detection System (IDS) tem, como finalidade, inspecionar o conteúdo do tráfego de rede e localizar possíveis

[2]**Intruso Interno -** Todos os utilizadores que tem algum tipo de autorização de acesso legitimo ao sistema e que, tentam realizar ações de intrusão a esse mesmo sistema.

[3]**Intruso Externo -** Todos os utilizadores que não pertencem ao sistema e que implementam ações de intrusão ao mesmo.

ataques, como um software Antivirus (AV). Ele inspeciona o conteúdo transmitido a procura de assinaturas de vírus informáticos, padrões que correspondem a software malicioso conhecido ou padrões de comportamento considerado fora do normal.

O conceito de avaliação está relacionado com a ação e o efeito de avaliar, que permite assinalar, estimar, apreciar ou calcular o valor de algo. Uma avaliação é uma forma de quantificar algo. No entanto encontrar uma maneira que permita fazer uma avaliação de software malicioso com alguma coerência é um processo com grande nível de dificuldade.

Uma possibilidade é basear, por exemplo, no comportamento do *malware*, identificando software malicioso segundo ações típicas (Avira Operation, 2013).

Existem muitos métodos que são utilizados para identificar software malicioso. Estes podem ser a detecção dos vírus informáticos, como por exemplo *worms* e *trojans*, utilizando sua assinatura, com um software Antivirus (AV). Outra técnica passa pela detecção baseada no comportamento, identificando o software malicioso em ações típicas, como por exemplo a conexão não solicitada e dissimulado à Internet, o que costuma se o caso dos *trojans* e dos *keyloggers*.

Em 1991 os membros da Computer Antivirus Rearchers Organizatios (CARO), apresenta uma nomenclatura baseada nos vírus de computadores para seus produtos AV. Alguns autores como por exemplo Szor (2005), utilizam basicamente a nomenclatura que é apresentada a seguir para definir os diversos tipos de códigos maliciosos:

Virus Informático

Apareceram pela primeira vez na década de 80. Apresentavam um código simples, apenas causando ocasionalmente inconvenientes ou apenas mostrando informações ao utilizador (Frederick B. Cohen, 1984). Hoje é um tipo de *malware* que é desenvolvido por programadores que, da mesma forma como um vírus biológico, infecta um sistema, se auto-replica e tenta se espalhar para outros computadores que estejam ligados à mesma rede. Atualmente este tipo de código é considerado de alto risco, podendo causar danos

significativos nos sistemas informáticos das organizações.

O mecanismo de infecção difere de cada vírus, sendo o mais comum a inclusão do seu código em programas executáveis de forma que quando o programa infectado é executado, este executa também o programa viral, contaminando assim outros programas.

Os vírus podem ser divididos em duas grandes categorias (Mell et al., 2005): vírus compilados e vírus interpretados.

Vírus compilado: seu programa fonte é compilado e direcionado para um determinado tipo de sistema operativo.

Vírus interpretado: seu código fonte só pode ser executado por uma determinada aplicação ou serviço, como por exemplo as macros ou scripts.

Worms

São parecidos com os vírus, se auto-replicam de um equipamento infetado para outro automaticamente, além disso os *worms* tém a capacidade de disseminar rapidamente manipulando as vulnerabilidades de um sistema ou de um aplicativo instalado no equipamento. A grande ameaça deste tipo de código malicioso reside no facto de afetar a performance de um sistema, sobrecarregando serviços e gerando tráfego, que em alguns casos provoca ataques de negação de serviço (Denial of Setvice (DOS)).

Trojans

Trojans ou Cavalos de Tróia, como na mitologia grega, é um código malicioso que esconde uma ação não esperada, entrando em um sistema e abre uma porta de comunicação para uma possível invasão. Atualmente este tipo de código é disfarçado como uma aplicação normal, por exemplo um jogo, o que torna difícil sua identificação. Este tipo de código não se replica.

As funções mais comuns apresentadas por este código, passam por coletar informações, esconder sua presença e desativar softwares de segurança como AV, firewall, etc..

Spyware

É um código malicioso que recolhe informações do utilizador sem seu conhecimento. Diferem dos *trojans* por não terem o objetivo de controlar o equipamento infetado. Visam principalmente roubo de dados confidenciais dos utilizadores como por exemplo dados de contas bancárias, passwords de acesso e documentos.

Backdoor

É uma falha na segurança que pode existir em um aplicativo ou sistema que permite a invasão de pessoas não autorizadas, permitindo o controlo do equipamento. Geralmente possuem capacidades especiais:

Zombies esse termo vem da utilização de maquinas infectadas com *bots*, possuindo capacidades de propagação como um worm. São controladas por atacantes que compõem um sistema maior. Botnets são redes compostas por milhares de computadores zumbis, utilizando ataques coordenados, como Distribution Deny of Service (DDoS), envio de spams e phishing.

Remote Administration Tools (RAT) permite ao atacante ter acesso a um sistema quando desejar, tendo acesso às imagens de tela, teclas digitadas e ficheiros sendo possível ativar remotamente dispositivos como impressoras, webcams, microfones e colunas de som.

Rootkit

É um software, na maioria das vezes malicioso, que foi desenvolvido para esconder alguns processos ou métodos normais de deteção e permitir o acesso privilegiado a um sistema.

2.1 Ciclo de vida do *malware*

A criação de um software malicioso é igual a qualquer outra aplicação informática, utilizando uma metodologia própria no seu desenho e implementação. Esta metodologia de desenvolvimento denomina-se ciclo de vida e contém uma série de etapas sendo a criação do próprio *malware* o início desse ciclo.

Basicamente pode-se dizer que esse ciclo engloba as seguintes etapas:

Criação: Na maioria das vezes, a criação de software malicioso começa com uma proposta por parte de indivíduos mal intencionados com algum propósito ilícito. Geralmente o desenvolvimento desse código tem como objetivo explorar e aproveitar alguma vulnerabilidade existente no sistema invadido, com o propósito de obter lucros financeiros ou simplesmente causar prejuízos na instituição invadida.

Desenvolvimento: É errado acreditar que a criação de software malicioso requer conhecimento de linguagem de programação. Existem muitas ferramentas para criar programas capazes de realizar ataques ao sistema sem necessariamente ser um *expert* em programação.

Replicação: Uma vez finalizado o desenvolvimento, é utilizado algum método para a transmissão. Normalmente os mecanismos de propagação escolhidos refletem as formas de comunicação de massa. Entre os mais comuns está correio eletrónico e redes P2P. Quem desenvolve o programa não é necessariamente o mesmo que o propaga.

Transição de dados: Esta fase é uma das mais importantes e obedece sempre à essência do software malicioso. Esta transmissão pode ser ativada por um método condicional, quando o utilizador executa uma ação específica ou quando o relógio do equipamento marca uma data especial, etc., ou diretamente quando o *malware* começa a implantar suas funções quando a infeção é concluída.

A partir do momento que o código é posto em circulação para sua propagação, cabe aos programadores de antivírus a sua deteção antes que causem algum dano ao utilizador. Assim temos o seguinte ciclo:

Caracterização: É a partir deste momento que as empresas de antivírus detetam a propagação e as características do código para criar a sua assinatura (sua impressão digital) que serve como identificação única.

Deteção heurística: Esta consiste em examinar e fazer relatórios sobre os ficheiros que se comportam como vírus mas que não estão contaminados com qualquer código conhecido. A deteção heurística procura determinadas características no comportamento de uma aplicação e tenta determinar se é ou não um código malicioso.

Deteção por assinatura: Depois de identificar a ameaça, os programadores de antivírus analisam o código para encontrar um método de deteção adequado e atualizam a base de dados de assinaturas. Este processo deve ser rápido e fundamental para que o *malware* não consiga causar danos aos utilizadores.

Eliminação: Nesta última etapa, já deve estar à disposição do utilizador final todas as atualizações com as assinaturas correspondentes aos códigos maliciosos reconhecidos, para que se possa detetar e eliminar qualquer tipo de ação provocada pelo *malware*.

É de extrema importância uma caracterização aprofundada dos códigos maliciosos, não só para nos proteger de futuros ataques mas também para saber que tipo de providencias tomar se o nosso equipamento tiver sido infetado. Na secção 2.2 deste capítulo é apresentada uma taxonomia que utiliza as características que foram aqui mencionadas, ajudando a identificar os métodos que serão aplicados no Capítulo 3.

2.2 Taxonomia: principais características

O *malware* agrupa aplicações prejudiciais ao equipamento, seja para o seu funcionamento, capacidade ou desempenho. É importante ressalvar que a presença de *malware* não significa necessariamente a destruição do equipamento. De fato, estudos recentes indicam que até 8 de cada 10 computadores têm *malware* instalado e continuam com as suas operações "quase" normais, embora com alguma lentidão.

O vírus é geralmente um programa de desempenho automático, concebido para interromper o funcionamento do equipamento e em alguns casos, também a destruição de dados. Pode-se identificar a presença de um vírus quando há uma alteração no funcionamento considerado normal do equipamento, como lentidão, desaparecimento de ficheiros, ou interrupção autónoma do seu funcionamento. Vírus e outro *malware* podem vir escondidos ou encapsulados dentro de conteúdos descarregados da internet.

Worm é um tipo de código malicioso que consegue reproduzir-se ocupando espaço disponível no equipamento. Isto irá diminuir o desempenho do seu computador, sem consequências maiores, permitindo identificá-los ou pelo menos suspeitar da sua existência.

Os *trojans*, por sua vez, são artefatos que estão escondidos dentro de outros ficheiros (daí o seu nome, em alusão ao cavalo de Troia). Sua principal função é facilitar o roubo do conteúdo contido no equipamento como por exemplo senhas, ficheiros contendo dados pessoais, etc. Têm diferentes níveis de risco e são identificáveis recorrendo a programas antimalware ou antivírus.

Assim considera-se como principais características as capacidades de:

- transmissão ou receção pela rede;
- transmissão ou receção por correio eletrónico;
- encapsulamento em ficheiros descarregados de sites na rede;
- transmissão por redes sociais;

- se esconder em mensagens instantâneas;
- explorar vulnerabilidades do sistema;
- clonagem;
- apagar ficheiros;
- gastar recursos do sistema;
- roubar informações.

O conjunto de informações reunidas podem ser resumidas na tabela 2.1. Essas informações podem ser utilizadas como referências para identificação do tipo de código malicioso em circulação na rede.

Tabela 2.1: Resumo comparativo entre os códigos maliciosos.
Adaptada Centro de Estudos (2012)

	Códigos Maliciosos						
	Vírus	**Worm**	**Bot**	**Trojan**	**Spyware**	**Backdoor**	**Rootkit**
Como é obtido:							
Recebido automaticamente pela rede		X	X				
Recebido por e-mail	X	X	X	X	X		
Baixado de sites na Internet	X	X	X	X	X		
Compartilhamento de arquivos	X	X	X	X	X		
Uso de mídias removíveis infectadas	X	X	X	X	X		
Redes sociais	X	X	X	X	X		
Mensagens instantâneas	X	X	X	X	X		
Inserido por um invasor		X	X	X	X	X	X
Ação de outro código malicioso		X	X	X	X	X	X
Como ocorre a instalação:							
Execução de um arquivo infectado	X						
Execução explícita do código malicioso		X	X	X	X		
Via execução de outro código malicioso						X	X
Exploração de vulnerabilidades		X	X			X	X
Como se propaga:							
Insere cópia de si próprio em arquivos	X						
Envia cópia de si próprio automaticamente pela rede		X	X				
Envia cópia de si próprio automaticamente por e-mail		X	X				
Não se propaga				X	X	X	X
Ações maliciosas mais comuns:							
Altera e/ou remove arquivos	X			X			X
Consome grande quantidade de recursos		X	X				
Furta informações sensíveis			X	X	X		
Instala outros códigos maliciosos		X	X	X			X
Possibilita o retorno do invasor						X	X
Envia spam e phishing			X				
Desfere ataques na Internet		X	X				
Procura se manter escondido	X				X	X	X

Sistemas Alvo do *malware*

À medida que o avanço da tecnologia de dispositivos móveis e redes informáticas sem fios se vão tornando cada vez mais populares, os inumeráveis perigos e riscos que os acompanham aumentam igualmente. Qualquer sistema, seja ele móvel ou fixo, que utilize sistema operativo é ou pode ser alvo de ataques efetuados por *malware*.

De acordo com a Trend Micro Incorporated (2013), os números de ataques e roubos de informações pessoais e financeiras de pequenas empresas têm sido ampliados. Só no segundo trimestre de 2012 a companhia bloqueou mais de 140 milhões de ameaças dirigidas a esse público. O estudo ainda revela que 25 mil aplicativos maliciosos para Android foram identificados nesse trimestre, um aumento de 317 % em relação ao número de amostras encontradas no semestre anterior, com o agravante de que, segundo a empresa, apenas um em cada cinco dispositivos Android conta com aplicativos de segurança instalados.

Métodos de propagação

Os métodos de propagação que a maioria do software maliciosos utilizam são normalmente técnicas de engenharia social, explorando uma série de temas que agucem a curiosidade do utilizador comum. Além disso também são utilizados engodo, correio eletrónico normalmente com o intuito de roubar dados pessoais e credenciais de acesso entre outros.

2.3 Métodos de detecção

Qualquer software é para o computador, um ficheiro que contém instruções que serão armazenadas na memória e utilizadas pelo processador. Essas instruções são as chamadas que o software faz ao sistema quando são executadas. O computador e outros periféricos possui uma linguagem binária composta apenas por 0 e 1. Como o computador apenas compreende o binário, as instruções do software também são representadas em binário.

Desta forma, analisar determinado software acaba sendo por ser uma tarefa complicada. Existem alguns métodos que podem ser utilizados na detecção de *malware* que podem ou não ser utilizados em conjunto. Atualmente é comum os fabricantes de antivirus utilizarem uma tecnologia multicamada, onde os ficheiros são processados e, quando desnecessário a análise do ficheiro, é excluída da camada seguinte permitindo uma verificação mais rápida.

Detecção baseada em assinaturas

Este método tenta fazer a correspondência de ficheiros com assinaturas de *malware* conhecidas que é uma sequência de bytes característica de um software malicioso específico. É necessário uma análise detalhada para identificar a infeção exata.

Detecção baseada em polimórficos

É utilizado para determinar novas variantes de software malicioso reconhecidos, mesmo se o comportamento da nova variante for diferente. Este método é particularmente eficaz na deteção de software que utiliza macros e *scripts*.

Análise baseada em heurística

A análise heurística observa como um software se comporta, desta forma pode identificar se ele é ou não malicioso. Isso permite a detecção de software malicioso desconhecido.

Análise baseada em comportamento

Esta tecnologia analisa a forma como o software se comporta para determinar se o comportamento do ficheiro é hostil e impedir sua execução.

No próximo capítulo ir-se-á abordar os métodos utilizados na classificação do tráfego de rede para identificarmos anomalias baseadas nestas informações.

Capítulo 3

Métodos de Classificação de Tráfego de Rede

Em 2011 a Cisco publica um artigo onde diz que a utilização da internet iria quadruplicar até 2015, impulsionada principalmente pelo número crescente de utilizadores de dispositivos móveis (tablets e smartphones). O aumento da largura de banda permitindo cada vez maiores velocidades e a facilidade de acesso são contributos relevantes para esse crescimento. Neste cenário é fundamental criar metodologias que permitam uma correta classificação do tráfego de rede. Muitas tarefas de gestão de rede, como por exemplo a caracterização da carga de trabalho, planeamento da capacidade, provisão de rotas, modelação e monitoração do tráfego de rede depende da identificação e classificação desse mesmo tráfego.

Existe hoje um número considerável de aplicações que utilizam a internet e esse número tem vindo a aumentar de forma progressiva. Aplicações multimédia, redes sociais, programas de Enterprise Resource Planning (ERP) e Customer Relationship Management (CRM), entre outros, têm se tornado essenciais no mundo dos negócios tanto a nível particular como empresarial. Nesse sentido, anomalias no tráfego de rede podem causar

graves problemas no seu funcionamento, sendo uma das principais preocupações para entidades que dependem da transmissão de dados para os seus negócios. A identificação e eliminação dessas anomalias deve ser feita o mais rápido possível ou corre-se o risco de prejuízos elevados em consequência dessas anomalias.

Modificações nas características da rede tais como o aumento da quantidade transmitida de pacotes, é um dos tipos específicos de anomalias e pode ser causado por vários fatores, como por exemplo, problemas físicos ou técnicos na rede, como cortes de energia ou configurações inadequadas dos equipamentos, até comportamentos maliciosos intencionais, como ataques de software malicioso ou tentativas de acesso não autorizado. Por isso é fundamental encontrar métodos de classificação de tráfego de rede eficazes, afim de assegurar um serviço de qualidade e ao mesmo tempo seguro.

O problema principal é lidar com a análise do tráfego de uma rede em larga escala. Técnicas de captura de pacotes e análise de seus conteúdos, são geralmente utilizados para gerar dados estatísticos e detectar eventos anómalos. Mas em uma rede de grande porte que possui uma quantidade de dispositivos interligados considerável, esta técnica torna-se inviável, pelo facto de ser necessário uma grande capacidade de processamento e espaço para armazenar os dados. As técnicas de classificação tradicionais, tais como as que se baseiam nos números de portas de comunicação ou em análise do *payload* dos pacotes, também não são eficazes para todos os tipos de tráfego de rede (Zavala et al., 2013).

Devido à sua natureza e aos fundamentos de muitas outras técnicas, o campo de qualificação do tráfego de rede tem mantido um grande interesse por parte de muitos autores. Por exemplo Moore and Zuev (2005) menciona em seu trabalho que uma técnica comum para identificar aplicações de rede na Internet é através da monitorização do tráfego baseado na utilização de portas de comunicação mais conhecidas, fazendo uma análise dos cabeçalhos dos pacotes para identificar tráfego que está associado a uma porta em particular e, portanto, de uma aplicação em particular (Moore et al., 2001). Esse processo pode levar a estimativas imprecisas devido a quantidade de tráfego realizado pelas mais diversas aplicações dado que os protocolos como o HyperText Transfer Protocol Secure

(HTTPS), são frequentemente utilizados para retransmitir outros tipos de tráfego, por exemplo, uma Virtual Local Area Network (VLAN) sobre HyperText Transfer Protocol (HTTP). Por outro lado, algumas aplicações ponto-a-ponto evitam utilizar portas de comunicação conhecidas, na tentativa de iludir algum bloqueio imposto pelo sistema, o que pode causar ainda mais imprecisão nas estimativas.

Analisar o tráfego capturado, encontrar padrões similares que possam ser utilizados para a classificação de alguma anomalia, permite criar uma base com dados de comparação com a informação resultante dessa análise. Lakhina et al. (2005) refere o conceito de deteção de anomalias utilizando entropia, onde é citado a entropia de Shannon. Este tipo de deteção e classificação é feito através do cálculo de entropia para as seguintes categorias: porta de origem, porta de destino, endereço IP, origem e endereço IP, destino, a partir da correlação entre si. Pode-se comparar valores de entropia para classificar comportamentos anómalos no tráfego analisado (M.L.Monsores et al., 2012).

Durante a pesquisa realizada para este trabalho, detetou-se que existe uma relação entre a classe de tráfego e suas propriedades estatísticas que fornecem informações sobre a distribuição do fluxo de bytes de pacotes e a saída de um número especifico de aplicações específicas. Observou-se ainda que a relevância dos pacotes utilizados nos estudos realizados por (Paxson and Floyd, 1995), tendem para um determinado tamanho de perfil, a observação desse perfil permite uma seleção mais precisa do tráfego relevante para esses estudos.

Um esquema de classificação deve permitir descrever padrões de tráfego representativos. A figura 3.1 mostra a nossa versão dos passos a seguir, que servem como guias para a realização desse processo. Na área da segurança, utilizar essa informação é crucial na identificação de anomalias no sistema de rede. O nosso objetivo passa por compreender como é feita a caraterização e deteção de anomalias no tráfego de rede. Este capítulo tem a finalidade de explicar algumas das técnicas mais utilizadas na classificação do tráfego de rede, procurando respostas viáveis para o cenário atual. O objetivo é identificar um comportamento fora do normal que indicie uma invasão ou utilização de software malicioso

durante o funcionamento normal do sistema.

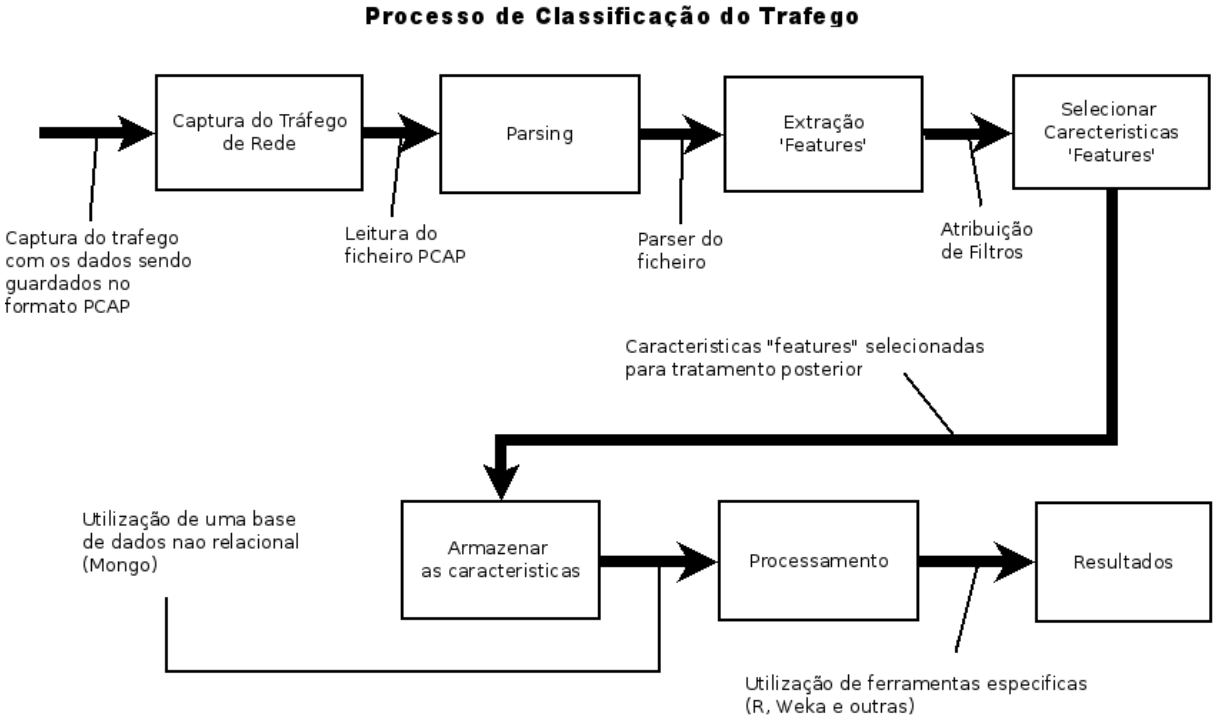


Figura 3.1: Passos a seguir para a classificação do tráfego

3.1 Conceitos Gerais

Antes de começar, são necessários alguns conceitos para a compreensão do conteúdo deste trabalho. Esses conceitos passam por saber como se processa o tráfego na rede, o que são pacotes de rede e utilização de máquinas virtuais para análise forense.

O tráfego de rede é a quantidade de informação que é trocada entre um servidor e os equipamentos que acedem a esse servidor e vice-versa. Segundo Forouzan (2006), um sistema básico de transmissão de informação é composto de cinco elementos (figura 3.2).

1. **Mensagem:** a mensagem é a informação (dados) a ser transmitida. Pode ser constituída de texto, números, figuras, áudio ou video - ou qualquer combinação desses.

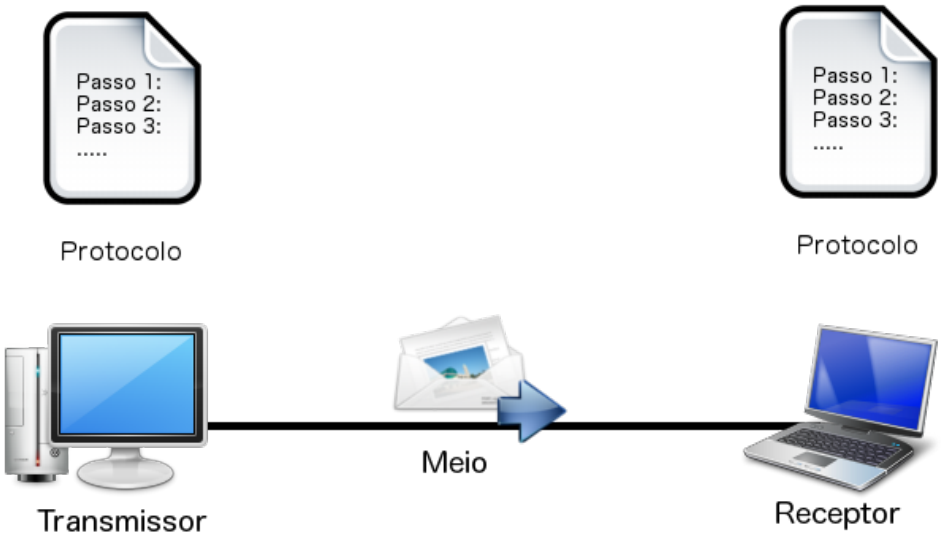


Figura 3.2: Cinco componentes da comunicação

2. **Transmissor:** o transmissor é o dispositivo que envia a mensagem de dados. Este pode ser um computador, uma estação de trabalho, uma câmera de video, um equipamento móvel (tablets ou smartphones) e assim por diante.
3. **Receptor:** o receptor é o dispositivo que recebe a mensagem. Como o transmissor pode ser um número variável de equipamentos.
4. **Meio:** o meio de transmissão é o caminho físico por onde viaja uma mensagem originada no transmissor e dirigida ao receptor. Pode ser por cabo, fibra óptica ou ondas de rádio (micro-ondas terrestre ou via satélite).
5. **Protocolo:** um protocolo é um conjunto de regras que governa a comunicação de dados. Ele representa um acordo entre os dispositivos que comunicam. Sem um protocolo, dois dispositivos podem estar conectados, mas sem comunicação entre si. Por exemplo, uma pessoa que fala apenas o francês dificilmente compreenderá o que diz outra pessoa que só fala japonês.

Tráfego de rede representa uma sequência de pacotes com a mesma assinatura em uma rede. Essa assinatura contém as seguintes informações:

- Origem IP address;
- Destino IP address;
- Porto da origem (por exemplo User Datagram Protocol (UDP) ou TCP port);
- Porto do destino (por exemplo UDP ou TCP port);
- Interface de entrada no elemento da rede;
- Interface de saída no elemento da rede;
- Protocolo nível 4 modelo Open Systems Interconnection (OSI), TCP, UDP, Internet Control Message Protocol (ICMP) , etc.);
- Classe de serviço representada pelo byte Type of Service (ToS) do cabeçalho IP.

Esta informação significa que os pacotes IP com essas informações em comum são agrupados no tráfego da rede, que oferece contadores de pacotes, bytes e registos das horas do início (primeiro pacote) e do fim do fluxo (último pacote ou timeout), sistema autónomo de origem e destino, etc. (figura 3.3). Em redes informáticas, um pacote, trama ou datagrama é um conjunto de dados formatados e transmitidos por uma rede informática . A informação a transmitir é dividida em inúmeros pacotes e enviada. O pacote possui um cabeçalho, que contém informações tais como, endereço do destinatário, soma para verificação de erros, prioridades, entre outras.

Ligações de comunicação entre equipamentos que não suportam tais pacotes, como comunicação ponto-a-ponto, simplesmente transmite uma série de bytes, caracteres ou bits sozinhos. Quando os dados são formatados em pacotes, os dados entre a comunicação podem ser compartilhados entre os utilizadores da rede. A figura 3.4 ilustra a estrutura do cabeçalho IP (Cisco, 2011a).

3.2 Máquinas virtuais para análise forense

Em ciências da computação, máquina virtual é o nome dado a uma máquina, implementada através de software, que executa programas como um computador real, esse processo

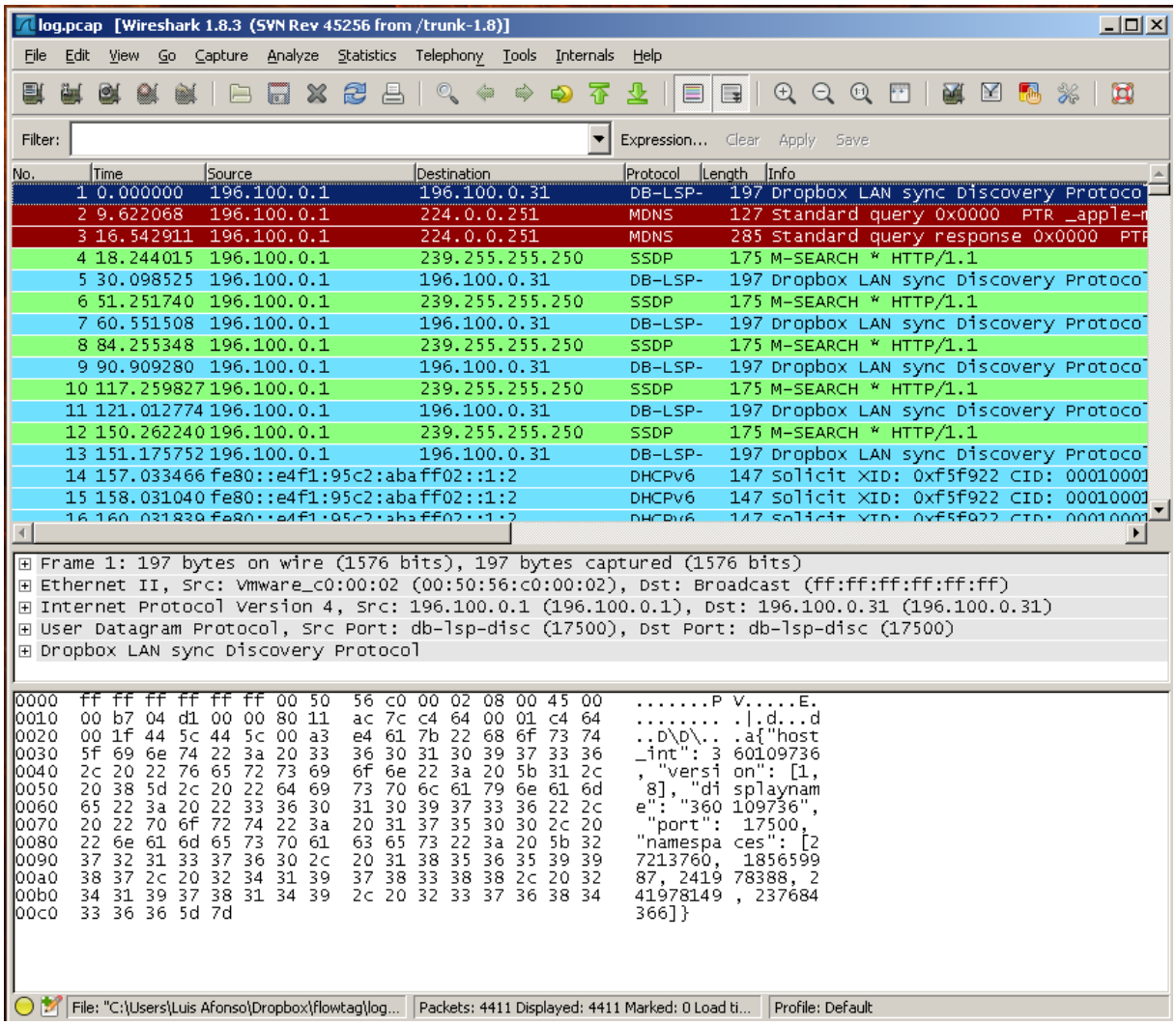


Figura 3.3: Tráfego capturado pelo *wireshark*

também é conhecido como virtualização.

A utilização de máquinas virtuais para análise de código malicioso tem sido aplicada por vários laboratórios de análise forense por ser um ambiente controlado. A informática forense compreende a aquisição, preservação, identificação, extração, restauração, análise e documentação de evidências informáticas, que sejam componentes físicos ou dados que foram processados eletronicamente e guardados em dispositivos de armazenamento (Kruse and Heiser, 2002).

Um incidente de segurança pode ser causado por um qualquer tipo de ataque realizado

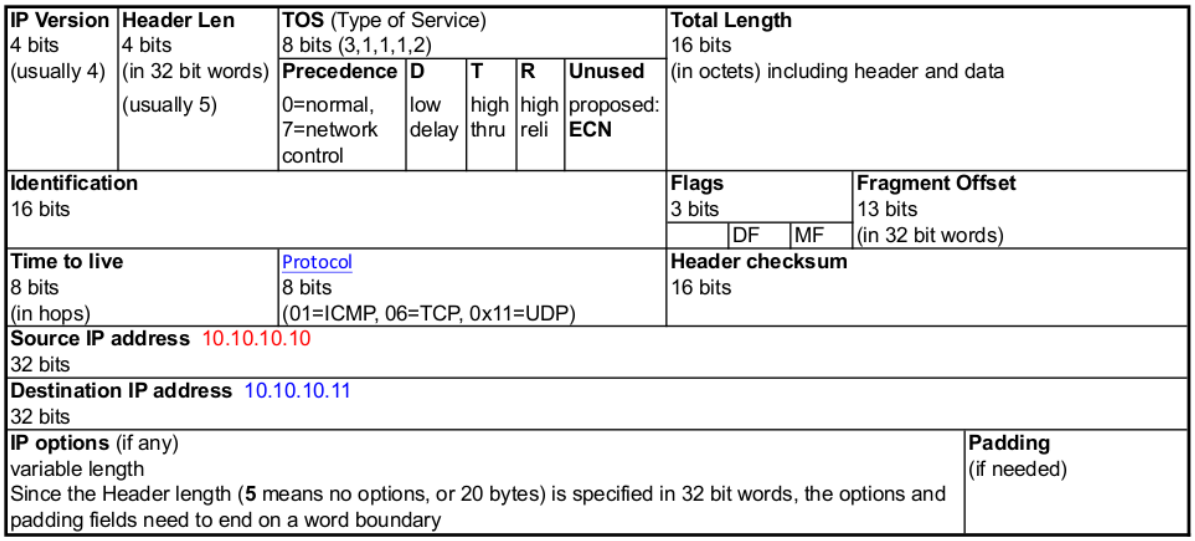


Figura 3.4: IP packet header

à infra-estrutura de informação de uma entidade, seja ele um ataque lógico ou um ataque físico. Os tipos mais comuns são os ataques lógicos, como ataques DOS, roubo ou perda de informação e infeção por vírus (Venere, 2009).

Segundo Venere, a resposta a um incidente de segurança está dividida em seis passos:

- Preparação - A entidade define os ativos que deseja proteger e as medidas cabíveis em caso de incidentes.
- Identificação - São utilizados mecanismos e técnicas para identificar a ocorrência de um incidente e definir a extensão do ataque.
- Contenção - Deve-se conter o ataque, impedindo que o invasor consiga acesso a outros sistemas e minimizando a atividade do atacante.
- Erradicação - As ações do atacante devem ser erradicadas com a aplicação de filtros e impedindo qualquer atividade do invasor na rede.
- Recuperação - Os sistemas invadidos devem ser recuperados e reinstalados.
- Acompanhamento - As técnicas e vulnerabilidades utilizadas pelo atacante devem ser estudadas e medidas devem ser aplicadas para impedir que elas funcionem.

3.3 Classificadores de tráfego

Conforme já mencionado, a aplicação de um sistema de classificação exige parametrizações dos objetos que serão classificados. A utilização desses parâmetros permite atribuir um determinado objeto a uma classe.

O fluxo de tráfego que é representado por um ou mais pacotes de uma determinada transmissão, utilizado como objeto na maioria das abordagens sobre a matéria, é definido pelos endereços IPs origem e destino, o tipo de protocolo (ICMP, TCP ou UDP) e, no caso de UDP e TCP, os números das portas utilizadas pelos dois hosts. No caso do TCP, um fluxo tem uma duração finita definida pela semântica do protocolo TCP. Alguns autores fazem uma classificação do tráfego padrão da rede TCP/IP gerado a partir de serviços Web (HTTP) onde cada registo de secção é descrito por nove atributos (Tabela 3.1). Esses atributos são utilizados para treinar classificadores onde são conseguidos resultados satisfatórios na deteção de anomalias de um modo rápido e preciso.

Tabela 3.1: Atributos tipicamente utilizados em deteção de anomalias

Atributo	**Descrição**
Duração	Duração da conexão
Src_Ip	IP de origem
Src_Bytes	número de bytes enviados
Num_root	número de acessos "root"
Num_access_files	número de aplicações de controlo de acesso a ficheiros
Num_shells	número de prompts de shells
Service	Acesso de serviço por porta: HTTP, FTP....
Count	número de conexões para o mesmo destino na mesma conexão nos últimos 2 sec.
Serror_rate	% de conexões com erros "SYN"
Same_srv_rate	% de conexões do mesmos serviço

Existem alguns modelos de classificadores utilizados no processo de classificação do tráfego de rede. Nas seções seguintes vamos falar sobre esses modelos.

3.3.1 Árvore de Decisão

A árvore de decisão é uma estrutura de dados, sendo percorrida da raiz até uma folha. Cada nó interno ou nó de decisão especifica um atributo de teste. Para cada resultado existe uma ligação para uma subárvore. Cada subárvore possui a mesma estrutura que a árvore (LIMA et al., 2010).

Segundo Breiman et al. (1984), árvores de decisão são modelos utilizados em problemas de aprendizagem de máquinas, em que o referido modelo é induzido a partir de um conjunto de instâncias de treino, com as classes previamente conhecidas. Cada instância é composta por um conjunto de atributos de entrada que são utilizados para classificar uma classe em particular, indicada por um atributo objetivo.

O algoritmo ID3 (Quinlan, 1986) constitui uma das referências base para os algoritmos atuais sobre árvores de decisão. Foi desenvolvido para o tratamento de problemas com características discretas, com uma estrutura básica, muito simples no que diz respeito ao tratamento de problemas sem erro de classificação nos dados. A construção da árvore é feita de cima para baixo, com o objetivo de escolher o melhor valor para cada um dos nós. É atribuída uma medida estatística que é a base na construção da árvore. Esta medida estatística consiste em ter um conjunto de vários elementos S, e um conjunto de n classes $C = C_1, C_2, ..., C_n$, sendo p_i a probabilidade da classe C_i em S, então a entropia do conjunto S, é a homogeneidade deste, traduzida na seguinte igualdade:

$$Entropia = -\sum_{i=1}^{c} p_i \log_2 p_i \tag{3.1}$$

sendo p_i a probabilidade do ramo i.

3.3.2 Naive Bayes

Segundo Zhang (2004), Naive Bayes é um dos algoritmos de aprendizagem mais eficiente para a aprendizagem de máquina do tipo classificação. Baseado no teorema de Bayes de

Thomas Bayes (Bellhouse, 2004), a sua principal característica é assumir a independência entre os atributos dos dados.

Naive Bayes usa a probabilidade $P(c|d)$ de um determinado elemento pertencer a uma determinada classe a partir da probabilidade inicial $P(c)$ de um elemento pertencer a esta classe e das probabilidades condicionais $P(t_k|c)$ de cada termo t_k ocorrer em um elemento da mesma classe. O objetivo é encontrar a melhor classe C_{map} para um elemento maximizando a probabilidade conforme a Equação 3.2, onde n_d é o número de termos no elemento d (Lucca et al., 2013).

$$C_{map} = argmax_{c \in C} P(c|d) = argmax_{c \in C} P(c) \Pi_{1 \leq k \leq n_d} P(t_k|c) \tag{3.2}$$

3.3.3 Redes Bayesianas

Matematicamente, uma Rede Bayesiana é definida como uma representação compacta de uma tabela de conjunção de probabilidades de universo do problema (Marques and Dutra, 2003). Do ponto de vista de um especialista, Redes Bayesianas constituem um modelo gráfico que representa de forma simples as relações de causalidade das variáveis de um sistema.

Uma das suas principais características é a adaptabilidade, podendo, a partir de novas informações, e com base em informações verdadeiras, gerar alteração nas dependências e nos conceitos, permitindo desta forma que as probabilidades não sejam meros acasos (Equação 3.3).

$$\begin{aligned} p(x_1, \cdots, x_n) &= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots p(x_n|x_1, \cdots, x_{n-1}) \\ &= \prod_{i=1}^{n} p(x_i|x_1, \cdots, x_{i-1}) \end{aligned} \tag{3.3}$$

3.4 Algoritmo para classificação de anomalias no tráfego de rede

Esta secção, descreve um algoritmo que realiza a deteção e classificação de anomalias no tráfego de rede. Estas anomalias são definidas com um desvio acentuado em determinados pontos do tráfego, tendo em consideração o modelo de comportamento considerado normal, como já foi dito anteriormente. As técnicas mais recentes de deteção de anomalias tem evoluído ao ponto de identificar quais são os endereços IP e portas origem e destino, por exemplo, que causam a anomalia, além de identificar as variáveis que estão sendo afetadas por esta. Contudo, esta informação, apesar de ser um processo considerável (Raphanelli, 2008), ainda não é suficiente para que os administradores, consigam lidar com o grande número de anomalias existentes no tráfego. A automatização da classificação do tráfego pode auxiliar os administradores com esse problema. O ponto principal da deteção de anomalias é proporcionar o maior número de informações, sobre as anomalias detetadas, com o intuito de disponibilizar uma base de decisão, dando prioridades a anomalias detetadas manualmente pelos administradores e, em alguns casos, permitir estratégias automáticas de tratamento dessas anomalias. Trabalhos realizados nesta área, mostram que, a utilização de métricas de volume e distribuição de endereços IP e portas, não são suficientes para distinguir entre os diferentes tipos de anomalias sem ambiguidades. A partir dessa ideia, é possível desenvolver um algoritmo que além de detetar a anomalia, obtém informações mais especificas sobre cada anomalia, tendo a capacidade de classifica-las com maior segurança. Para atender a essas necessidades, o algoritmo apresentado nesta secção, possui quatro etapas bem definidas:

1. realizar a deteção de anomalias utilizando múltiplas características encontradas no tráfego;
2. para cada anomalia identificar os pacotes responsáveis;
3. utilizar esses pacotes para derivar novas características especificas da anomalia;

4. classificar a anomalia utilizando estas características e criar um módulo de classificação baseado em regras.

3.4.1 Primeira etapa: deteção de anomalias

Essa parte do algoritmo analisa o tráfego, encarregando-se da deteção das anomalias. O algoritmo aqui apresentado baseia-se no Network Anomaly Detection Algorithim (NADA) (Farraposo et al., 2007a), desenvolvido no contexto do projeto metrosec, para detetar e caracterizar anomalias do tráfego de rede. Este algoritmo é utilizado para detetar qualquer anomalia responsável por algum nível de variação em ao menos um dos critérios utilizados em alguma escala de tempo e em algum nível de agregação IP.

A equação 3.4 representa como funciona o algoritmo NADA. Esta baseia-se no conceito de deltóides absolutos, desenvolvido por Cormode and Muthukrishnan (2005), para detetar mudanças significativas na variação de um parâmetro.

$$
\begin{aligned}
X &= \{x_1, x_2, ..., x_n\}, & x_i &= \{\#packets|\#bytes|\#syn\}/\Delta \\
P &= \{p_1, p_2, ..., p_{n-1}\}, & p_i &= x_{i+1} - x_i \\
&\begin{cases} p_i \geq K * \sigma p & \text{anomalous} \\ p_i < K * \sigma p & \text{not anomalous} \end{cases}
\end{aligned}
\tag{3.4}
$$

A equação 3.4 pode ser explicada da seguinte forma. Dado o ficheiro de captura de tempo T, com um intervalo de escala de tempo delta (30 segundos), divide-se os dados em N slots[4] (secções de tempo), em que N pertence $[1, T/\Delta]$. Para cada slot i, obter a série de tempo X para cada característica considerada. Calcular os deltóides absolutos P de X e calcular o seu desvio padrão. Para qualquer p_i sobre o valor limite $K * \sigma p$ marcar o seu slot como anómalo.

[4] **Slot** é um termo em inglês para designar um encaixe ou espaço

No NADA, a equação é aplicada de forma recursiva. Cada nível de interação utiliza um nível de agregação do espaço IP diferente no algoritmo para a deteção, utilizando deltóides absolutos de séries de tempo de características do volume do tráfego de rede, constituindo um modelo estatístico para o comportamento padrão do tráfego de rede. Qualquer deltóide que desviar um determinado valor desse modelo, neste caso K desvios padrões, é considerado anómalo.

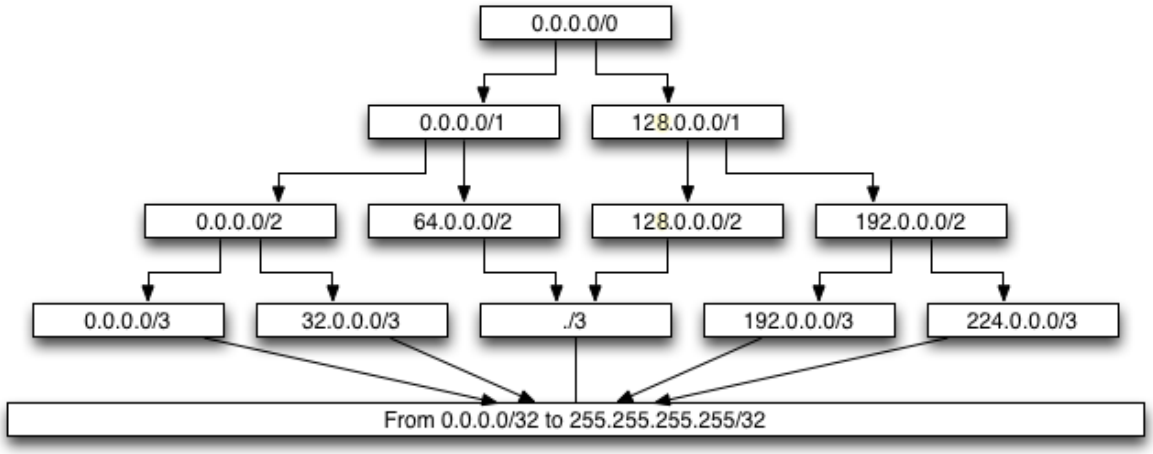


Figura 3.5: Níveis de agregação de endereços IP

A característica número de SYN é uma aproximação para a característica comummente utilizada de número de fluxos. Esta consiste no número de pacotes TCP que possuem apenas a flag SYN habilitada e diminuem consideravelmente o processamento necessário quando os dados estão no formato nativo. Este algoritmo é baseado nos níveis de agregação e é definido pela quantidade de bits do endereçamento IP de destino considerados. A figura 3.5 ilustra como os níveis de agregação estão divididos no espaço IP. Por exemplo, o nível 1 possui dois grupos, sendo que todos os endereços IP de destino com o primeiro octeto entre 0 e 127 estarão no primeiro grupo, e os demais no segundo. O nível 0 é o equivalente a agregação total do tráfego e é sempre considerado.

Farraposo et al. (2007b) utiliza um método que permite a identificação dos pacotes responsáveis e o impacto causado por cada nível de agregação, mas apenas deteta anomalias que geram uma variação considerável na agregação do tráfego.

3.4.2 Segunda etapa: Identificação dos Pacotes

Durante o processo de investigação para este trabalho, nomeadamente o estudo da caracterização de anomalias no tráfego de rede, observam-se diferentes tipos de anomalias que podem afetar os parâmetros de quantificação e as características do tráfego, como os endereços IP e portas de comunicação. Isto demonstra que não é possível classificar estas anomalias de forma confiável utilizando apenas esta informação, sendo necessário identificar outras características relativos à anomalia.

Um processo de classificação fiável necessita de atributos significativos, que proporcionem informações suficientes e que possibilitem a distinção entre os diferentes tipos de anomalias.

Segundo Raphanelli (2008), neste processo, o primeiro passo na derivação destes atributos considerados relevantes, é encontrar os pacotes responsáveis pela anomalia. Uma anomalia detetada na etapa anterior é identificada pelo slot, rede e nível de agregação. Sabe-se também que métricas foram significativamente afetadas pela anomalia, quando p é considerado anómalo. Esta informação permite ler todos os pacotes no slot correspondente para aquela rede e nível de agregação, identificando os pacotes correspondentes.

3.4.3 Terceira etapa: Derivação de atributos

Na primeira e segunda etapas desse processo, foi feita a extração dos principais atributos do tráfego de rede. Nesta etapa necessitamos fazer uma derivação mais aprofundada desses atributos, de forma a obter atributos específicos para cada anomalia. Já foi visto na secção 3.1, que atributos são considerados como relevantes para se ter informações fiáveis a respeito de anomalias no tráfego. De acordo com a anomalia encontrada, pode ou não ser necessária essa derivação.

3.4.4 Quarta etapa: Classificação

Esta é a etapa considerada mais importante deste processo para que os administradores de rede possam rentabilizar seu tempo de uma forma mais eficiente, obtendo assim mais informação sobre cada anomalia, permitindo uma priorização adequada para análise manual. Contudo, o objetivo de se conseguir uma classificação automatizada das anomalias no tráfego de rede com uma taxa reduzida de falsos positivos ainda é um problema em aberto e de difícil resolução. A grande quantidade de tipo de anomalias e as suas variações tornam necessária a criação de assinaturas extremamente especializadas para uma diminuição da taxa de falsos positivos.

No âmbito deste trabalho, foram identificados três tipos de assinaturas para a classificação automatizada: assinaturas universais, assinaturas fortes e assinaturas locais. Assinaturas universais utilizam regras que caso satisfeitas, não erram na classificação da anomalia. Normalmente apoiam-se nas especificações do protocolo, não são afetadas pelo tipo de rede e podem ser aplicadas universalmente. No caso das assinaturas fortes, estas proporcionam uma taxa de falsos positivos baixa que podem variar de acordo com as características da rede. Normalmente possuem um ou mais valores limites difíceis de serem definidos para determinados atributos. É necessário que a taxa de falsos positivos seja baixa nesta assinatura para que o tempo de análise manual seja o mais curto possível. As assinaturas locais são especificas para cada domínio administrativo, sendo definidas localmente pelos administradores de rede (Raphanelli, 2008).

Capítulo 4

Uma abordagem à deteção de malware

Com base no estudo descrito atrás, este capítulo apresenta o processo de desenvolvimento do trabalho realizado na vertente prática, utilizando o conhecimento adquirido.

4.1 Captura de tráfego

O objetivo deste exercício é apresentar um modelo que possa ser utilizado na classificação de tráfego anómalo encontrado numa rede de computadores gerado por *malware*. Para isto, ir-se-á utilizar informação extraída do tráfego, tentando encontrar padrões que possam ser utilizados como regras para identificar uma anomalia nesse mesmo tráfego. Foram realizados algumas etapas ao longo deste trabalho essenciais para a compreensão de metodologias aplicadas nesta área.

Primeiro é necessário capturar tráfego de rede que contenha informação útil, para ser utilizada com os métodos de classificação apresentados no capítulo 3. Este foi sem dúvida o processo mais complicado de todo o trabalho, visto que seria necessário um número de amostras grande em relação a quantidade de *malware* existente. Nesta etapa foi necessário

criar um cenário para captura de tráfego. Para esse efeito configurou-se um equipamento com software de virtualização que permitiu a simulação de uma rede de computadores, criando máquinas virtuais que permitiram a instalação de vários sistemas operativos com a possibilidade de executar em simultâneo aplicações que geram tráfego de rede num ambiente controlado. Esse computadores (virtuais) estão dentro de um computador físico com um sistema operativo totalmente distinto (Figura 4.1).

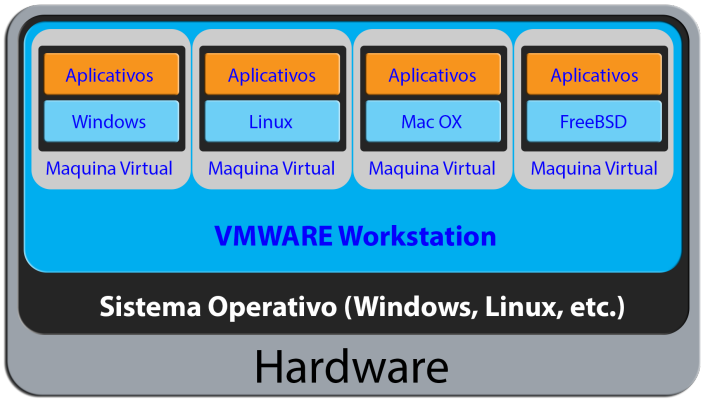


Figura 4.1: Esquema lógico utilizado pelo VMWARE.

O esquema apresentado na figura 4.2 mostra as máquinas virtuais em rede simulando uma intranet com configurações distintas, sendo que uma das máquinas foi infetada com vírus para estudo do seu comportamento. Outra máquina possui software de deteção de vírus que cria alertas quando encontra algo anormal na comunicação ou quando identifica alguma espécie de *malware*. Uma terceira máquina contém software que simula ataques e tentativas de intrusão. As restantes máquinas são utilizadas como referência para analisar o tráfego gerado pala máquina infetada.

Como já mencionado, foi utilizada virtualização para criar um laboratório de testes. Existem no mercado algumas ferramentas que são utilizadas para virtualização. Escolheu-se para este trabalho o VMWARE Workstataion. Este permite a instalação de um sistema operativo dentro de outro dando suporte real a este sistema. Isso permite que o sistema seja executado em simultâneo mas num ambiente isolado. Desta forma é possível simular uma rede de computadores com pouca ou nenhuma segurança para que intencionalmente

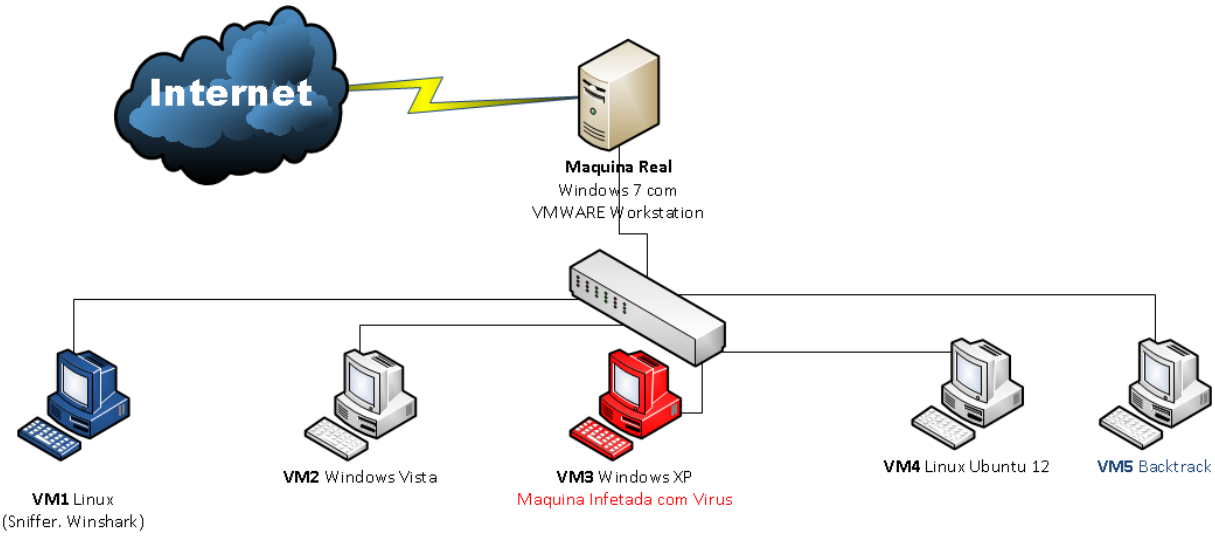


Figura 4.2: Diagrama do laboratório virtual

sejam utilizados para executar *malware*.

O equipamento de trabalho possui um processador Intel I7 e 6 Gb de memória Random Access Memory (RAM). Este fato limitou a utilização de várias máquinas virtuais trabalhando em simultâneo como inicialmente planeado. Assim foram criadas basicamente duas maquinas virtuais, uma contendo sistema operativo linux e o software de *sniffer*[5] *tcpdump*, *wireshark* e *snort*[6] e outra contendo uma versão do windows (Windows XP) como sistema operativo com o nível de segurança mínimo, permitindo ser infetada com *malware* responsável por tráfego anómalo.

Para que o *sniffer* consiga capturar os pacotes transmitidos entre equipamentos numa rede informática, é necessário que essa informação chegue à interface de rede do equipamento onde o *sniffer* está a ser executado[7]. Por essa razão, obrigatoriamente, toda a comunicação passa pela máquina que contém esse software.

[5]**Sniffer** é um software que tem como principal função à captura de pacotes transmitidos numa rede de computadores.

[6]O **Snort** é um software de livre utilizado para a deteção de intrusão em uma rede, capaz de analisar o trafego em tempo real.

[7]**Importante:**Numa rede que usa switches isso não acontece, então é necessário configurar uma span port para que todo o tráfego que passa nas outras portas do switch seja enviado também para a porta do equipamento *sniffer*.

Com o cenário montado e a funcionar, passamos a captura do tráfego deixando que a máquina que contém *malware* executasse livremente suas funções. Esse tráfego é capturado utilizando um *sniffer* que fica atento à comunicação entre as máquinas interna e externa para quando se ligada a internet. O *sniffer* guarda toda a informação transmitida e recebida nessa rede localmente (Figura 4.3, onde mais tarde essa informação será utilizada para a análise e classificação de tráfego rede.

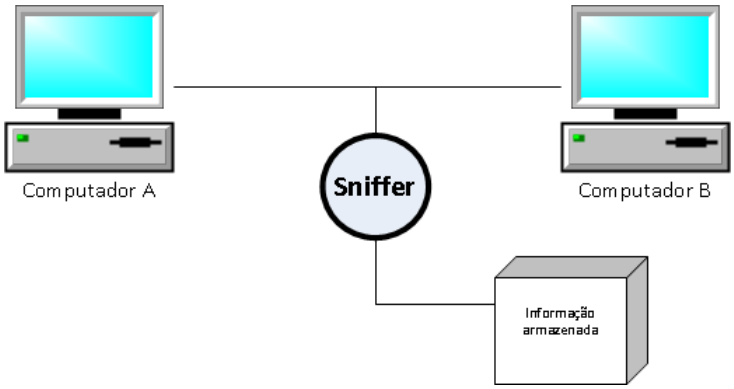


Figura 4.3: Esquema de utilização do software de *sniffer*

O objetivo da recolha de dados, como já foi dito, é constituir uma base de informação onde possamos aplicar técnicas de análise sugeridas por vários autores e com o conhecimento adquirido, desenvolver metodologias capazes de identificar anomalias. Existem ferramentas de software de utilização livre (*sniffers*) como por exemplo o *wireshark* ou *tcpdump* que interceptam e analisam o tráfego de uma rede. Estes programas são destinados à monitorização de informação do tráfego numa rede de computadores. A informação trocada entre os computadores, normalmente conhecida por pacotes (packets) tem uma determinada estrutura que é constituída por um cabeçalho que inclui entre outros elementos, os endereços do emissor e destinatário da mensagem que passam pela interface de rede. Os pacotes são constituídos também por uma parte de dados, conhecida como *payload*.

As funcionalidades do *wireshark* são parecidas com o *tcpdump* mas com uma interface gráfica, com mais informação e com a possibilidade da utilização de filtros. Isso nos permite controlar o tráfego da rede, ver qual a informação que entra e sai, permitindo

fazer uma análise detalhada, conseguindo com isso, encontrar padrões que podem ser classificados como anomalia ou identificando o próprio *malware* por assinatura.

Na figura 4.4 apresentamos o fluxo que demonstra o processo a seguir para classificar o tráfego capturado.

Figura 4.4: Fluxo de processamento

O *tcpdump* é simplesmente executado na linha de comandos da seguinte maneira:

```
1 # tcpdump
```

Desta forma o tráfego que passa pela primeira interface (excluindo a loopback, o que geralmente deixa a eth0) será mostrado com uma descrição rápida dos pacotes capturados. Por exemplo:

```
1 # tcpdump
2 tcpdump: data link type PKTAP
3 tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
4 listening on pktap, link-type PKTAP (Packet Tap), capture size 65535 bytes
5 12:45:31.826213 IP mbp-de-luis.home.64128 > antoniorasmbp15.home.afpovertcp: Flags [P
      .], seq 1454032914:1454032930, ack 3227850350, win 8192, options [nop,nop,TS val
      843728059 ecr 1090659618], length 16
6 12:45:32.001915 IP antoniorasmbp15.home.afpovertcp > mbp-de-luis.home.64128: Flags
      [.], ack 16, win 8191, options [nop,nop,TS val 1090673435 ecr 843728059], length 0
7 ....
```

A informação apresentada pelo *tcpdump* não é a mais aconselhada quando se faz uma análise mais profunda e principalmente quando se quer análisar os pacotes capturados. Sem contar que a informação capturada não está a ser armazenada, apenas sendo exibida e descartada.

Para o trabalho desenvolvido foram configurados alguns parâmetros do *tcpdump* necessários para resolver algumas situações. Em primeiro lugar ir-se-á especificar a interface na qual o *tcpdump* irá escutar, com a opção *-i eth#*. Isto fará com que o software esteja atento a toda a comunicação que passe pela placa de rede indicada (# é o número da placa de rede utilizada). O *tcpdump* tenta resolver os problemas de resolução de Domain Name System (DNS) antes de apresentar a informação recolhida, podendo ocorrer algum problema (por exemplo, um delay muito alto na rede). Isso pode provocar a não amostragem de alguns pacotes enquanto o programa aguarda a resolução do nome de algum IP. Para indicar ao *tcpdump* para não resolver qualquer problema com nome DNS, utiliza-se a opção *-n*. A opção *-nn* faz com que o *tcpdump* não tente resolver nomes de protocolos e porta. É importante guardar toda a informação recolhida em um ficheiro para ser analisado com o intuito de identificar alguma anomalia, para isso utilizamos a opção *-w [Nome do Ficheiro]*.

```
# tcpdump -nn -ni eth1 -w [nome do ficheiro de saida]
```

Com o comando anterior inicia-se a captura de tráfego e a informação é armazenada em ficheiros com o formato Packet CAPture (PCAP). Estes ficheiros constituem a base de dados para pré-processamento. O ficheiro PCAP é lido com auxilio de bibliotecas específicas desenvolvidas para diversas linguagens de programação como, por exemplo, C++, Ruby ou Python. O exemplo da utilização dessa biblioteca pode ser visto no código apresentado no anexo A.1 e o resultado é mostrado a seguir num exemplo da leitura de um ficheiro capturado:

```
Packet number 240:
 timestamp seconds.......: 1355862370
 timestamp micro seconds.: 96955

 successfully casted to ip pcaket

                    From.: 196.100.0.20
                      To.: 196.100.0.1
```

```
               more flag.: 64

            packet length: 40
   Protocol: TCP
packet number 240 is recived
   Src port: 49159
   Dst port: 2869

Packet number 241:
 timestamp seconds.......: 1355862370
 timestamp micro seconds.: 97183

 successfully casted to ip pcaket

                    From.: 196.100.0.20
                      To.: 196.100.0.1
               more flag.: 64

            packet length: 231
   Protocol: TCP
packet number 241 is recived
   Src port: 49159
   Dst port: 2869
   Payload (191 bytes):
00000   47 45 54 20 2f 75 70 6e  70 68 6f 73 74 2f 75 64    GET /upnphost/ud
00016   68 69 73 61 70 69 2e 64  6c 6c 3f 63 6f 6e 74 65    hisapi.dll?conte
00032   6e 74 3d 75 75 69 64 3a  38 32 61 37 34 33 31 30    nt=uuid:82a74310
00048   2d 63 61 61 64 2d 34 33  66 65 2d 61 39 34 63 2d    -caad-43fe-a94c-
00064   32 64 62 33 64 37 61 61  62 36 37 31 20 48 54 54    2db3d7aab671 HTT
00080   50 2f 31 2e 31 0d 0a 43  6f 6e 6e 65 63 74 69 6f    P/1.1..Connectio
...
```

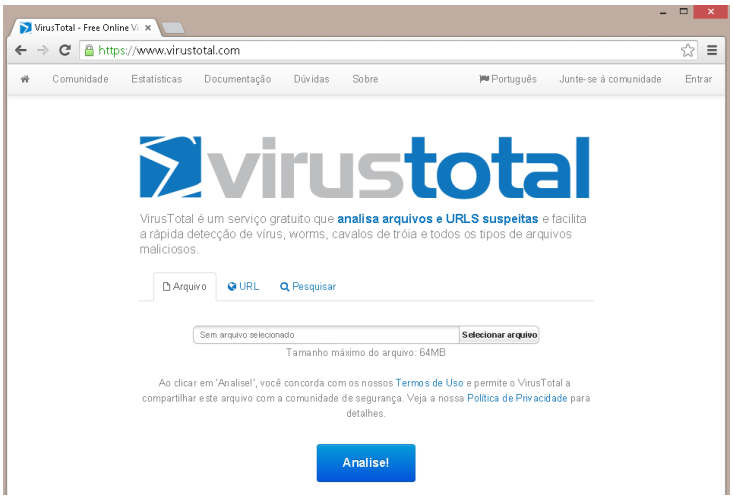


Figura 4.5: Página Virustotal

4.1.1 Pré processamento

Todo o pré processamento passa pela utilização de diversas ferramentas que ajudaram em todo o processo. Por exemplo, enquanto o *tcpdump* capturava o tráfego, o *snort* analisa o tráfego e gera alertas. Para análise e a verificação do conteúdo dos ficheiros PCAP utiliza-se uma ferramenta online conhecida como *virustotal*[8] (Figura 4.5).

No caso concreto dos ficheiros em análise obteve-se a resposta apresentada na figura 4.6, comprovando realmente que na informação transmitida durante esta captura ouve a tentativa de intrusão por *malware*, como se pode ver no excerto de um dos relatórios apresentando pela ferramenta.

```
1 PCAP file! The file being studied is a network traffic capture, when studying it with
      intrusion detection systems Snort triggered 54 alerts and Suricata triggered 92
      alerts.
2
3 Snort alerts
4     INDICATOR-SHELLCODE ssh CRC32 overflow filler (Executable Code was Detected)
5     POLICY-OTHER FTP anonymous login attempt (Misc activity)
```

[8]Virustotal é uma subsidiaria da Google, é um serviço gratuito online que analisa ficheiros e endereços URLs para a identificação de Virus, Worms, trojans e outros tipos de *malware* geralmente detetados por antvirus.

```
6 .....
7     MALWARE-CNC Win.Trojan.Ramdo variant outbound connection (A Network Trojan was
      Detected)
8 .....
```

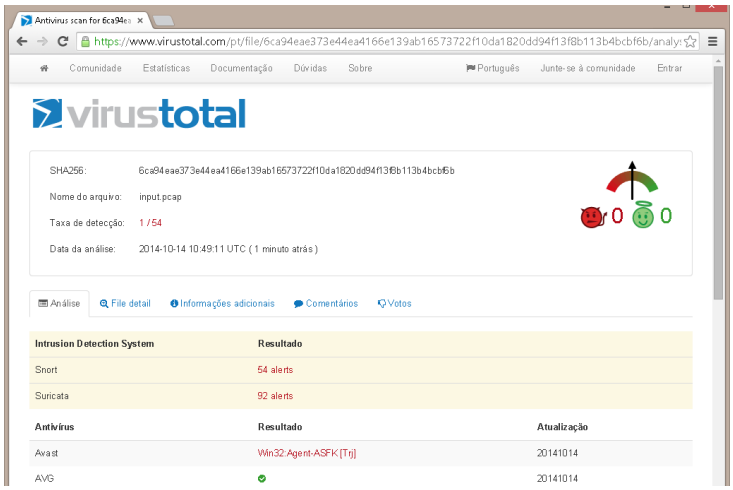


Figura 4.6: Resposta da análise do ficheiro capturado

Como já foi dito, o *snort* foi utilizado para criar um ficheiro de alertas, executando-se o programa da seguinte maneira:

```
1 # snort -c snort/etc/snort.conf -A fast
```

Desta forma o *snort* ficará atento ao tráfego da rede e todas as ocorrências durante a transmissão e recepção de dados será guardada. Essas ocorrências são guardadas em um ficheiro de texto em */var/log/snort/alert*. Por exemplo, se alguém executar um *portscan* no seu endereço IP, o *snort* ira escrever no ficheiro *alert* uma mensagem do tipo:

```
1 09/29-14:46:53.372042 [**] [1:1420:11] SNMP trap tcp [**] [Classification: Attempted
      Information Leak] [Priority: 2] {TCP} 192.168.1.100:39111 -> 192.168.1.2:162
2 09/29-14:46:53.375268 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification:
      Attempted Information Leak] [Priority: 2] {TCP} 192.168.1.100:39111 ->
      192.168.1.2:705
3 09/29-14:46:58.423072 [**] [1:1226:4] X11 xopen [**] [Classification: Unknown Traffic]
       [Priority: 3] {TCP} 192.168.1.100:41967 -> 192.168.1.2:6000
```

```
09/29-14:46:58.463005 [**] [1:1228:6] SCAN nmap XMAS [**] [Classification: Attempted
    Information Leak] [Priority: 2] {TCP} 192.168.1.100:39124 -> 192.168.1.2:1
```

Uma análise ao ficheiro *alert* gerado pelo *snort* durante a captura do tráfego forneceu as seguintes informação relativas ao *malware* encontrado:

```
[**] [1:2013076:6] ET TROJAN Zeus Bot GET to Google checking Internet connectivity
    [**]
[Classification: A Network Trojan was Detected] [Priority: 1]
10/07/12-14:37:18.666420 172.16.253.131:1985 -> 173.194.68.99:80
TCP TTL:128 TOS:0x0 ID:12501 IpLen:20 DgmLen:303
***A**** Seq: 0x29CA3699  Ack: 0x302F5B04  Win: 0xFAF0  TcpLen: 20
[Xref => http://lists.emergingthreats.net/pipermail/emerging-sigs/2010-October/009807.
    html][Xref => http://www.secureworks.com/research/threats/zeus/?threat=zeus]
```

Essas informações foram utilizadas para classificar o tráfego capturado. Para isso foi necessário criar uma rotina para ler o ficheiro com o tráfego capturado (PCAP) e o ficheiro *alert*, guardando o resultado um ficheiro contendo linhas de vetores classificadas como normal e anormal, utilizando a linguagem de programação RUBY. A listagem do código pode ser vista no anexo A.2.

Esse ficheiro tinha que obrigatoriamente ser padronizado de forma a ser utilizado em ferramentas especificas de tratamento de informação. Optou-se por gravar os ficheiros em formato Comma-separated values (CSV), formato simples e suportado pela maioria dos programas de folha de cálculo existentes no mercado. Isto permitiu especificar alguns atributos (tabela 4.1) que serão utilizados para analisar e classificar a informação recolhida no tráfego de rede.

Tabela 4.1: Atributos utilizados

Atributo	**Descrição**
packet_count	conta o numero de pacotes capturados
time	timestamp
eft	Ethernet Frame Type
hs	Header Size
tos	Type of Service
ps	Packet Size
id	Identificator
fl	Flags
fr	Fragment
pt	Protocol
sip	Source IP
sport	Source Port
dip	Destination IP
dport	Destination Port
udps	UDP Size
tcps	TCP Size
tcphs	TCP Header Size
idp	Protocol ID
ack	Acknowledge
win	Win
df	Fragment IP
tp	Tipe (Normal, Anomalo)

Exemplo da resultado:

```
1 No.,TIME,EFT,HS,TOS,PS,ID,FL,FR,PT,SIP,SPORT,DIP,DPORT,UDPS,TCPS,TCPHS,IDP,ACK,WIN,DF,
    TP
2 1,03:34:48.255531,1,5,16,74,5799,2,72603759,TCP
    ,192.168.0.2,1254,192.168.0.1,23,0,0,10,6,0,32120,1,normal
3 2,03:34:48.257221,1,5,0,74,29797,18,3225454542,TCP
    ,192.168.0.1,23,192.168.0.2,1254,0,0,10,6,72603760,17376,0,normal
4 ......
5 289,22:43:30.610247,1,5,0,342,4162,0,0,UDP
    ,172.16.16.128,54043,172.16.16.150,42283,308,0,0,17,0,NULL,0,anomalo
6 290,22:43:30.637253,1,5,0,66,21706,194,2538380055,TCP
    ,172.16.16.128,53948,172.16.16.150,135,0,0,8,6,0,3,0,normal
```

```
291,22:43:30.637888,1,5,0,66,11408,18,879752738,TCP
    ,172.16.16.150,135,172.16.16.128,53948,0,0,8,6,2538380056,8192,1,normal
.......
298,22:43:30.812267,1,5,0,342,4162,0,0,UDP
    ,172.16.16.128,54043,172.16.16.150,42283,308,0,0,17,0,NULL,0,anomalo
.......
```

4.2 Análise e classificação

O objetivo é ganhar uma compreensão de como o *malware* interage em uma rede informática. Com a compreensão do comportamento do *malware*, pretende-se criar regras de defesa que possam ser utilizadas nos sistemas para garantir sua proteção.

No capítulo 2 deste documento apresentaram-se conceitos que nos ajudam a identificar e classificar cada tipo de *malware*, mediante algumas características específicas de cada exemplar identificado. Apesar de cada investigador tratar de forma particular seu trabalho nesta área existe sempre pontos em comum. Neste trabalho utiliza-se a informação bibliográfica consultada e seguiu-se uma metodologia específica.

Durante o processo de investigação, a análise do *malware* possibilitou a aquisição de habilidades práticas com abordagem de metodologias diversas e a utilização de várias ferramentas de monitorização de redes informáticas, entre outras.

A classificação consiste em prever o tipo a que cada pacote capturado pertence sendo considerado *normal* ou *anómalo*. Esta classificação é feita localizando regras que dividam os dados indicados em grupos.

Neste ponto existe já um ficheiro classificado através da análise do ficheiro PCAP e o ficheiro de alertas criado pelo *snort* que será utilizado para treino. Foi criado um segundo ficheiro com os mesmos procedimentos utilizado para testes.

4.2.1 Aprendizagem

Para realizar a aprendizagem, recorreu-se à ferramenta WEKA (figura 4.7) que possui uma coleção de algoritmos muito utilizados na resolução de problemas de associação, classificação, regressão e clustering. Esta ferramenta foi desenvolvida pela universidade de Waikato e é gratuita e de código aberto. Este software é muito utilizado em aplicações e teste de Redes Neurais Artificiais (RNA)[9]

Figura 4.7: Interface inicial do WEKA

A interface gráfica do WEKA permite a execução dos algoritmos de data mining de forma interativa.

Iniciou-se o WEKA no modo de exploração, onde foi utilizado um dos ficheiros para análise infetados. O ficheiro possui um conjunto de dados com 4280 amostras das quais 19 delas são consideradas tráfego anómalo (Figura 4.8).

Como já referido anteriormente, o WEKA possui um vasto número de algoritmos de classificação. Pode-se trabalhar com árvores binárias, Support Vector Machines (SVM), ou construir algoritmos de classificação. Este processo ajuda a criação de um guião para a determinação de saída de uma nova instância de dados. Neste caso uma instância que nos diga se um determinado pacote na comunicação é considerado ou não anómalo.

[9]As **RNAs** são modelos computacionais inspirados pelo sistema nervoso central de um animal que é capaz de realizar uma aprendizagem da maquina bem como reconhecimento de padrões.

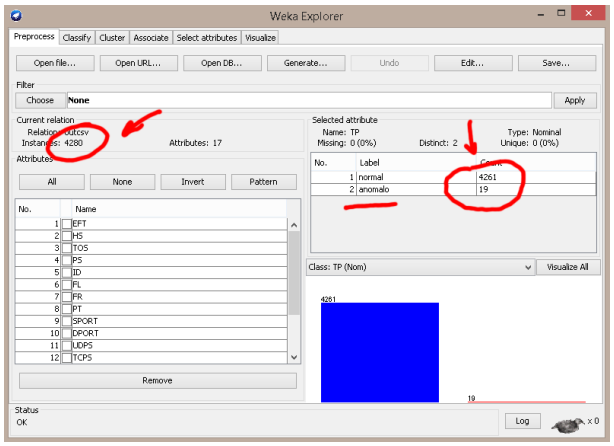


Figura 4.8: Interface inicial do WEKA

Os testes foram realizados com os classificadores J48, Naive Bayes e SVM e o tempo de resposta para o mesmo ficheiro apresentado na Tabela 4.2 refere-se apenas a aprendizagem.

Tabela 4.2: Tempos de construção de cada modelo

Classificador	**Tempo gasto na construção do modelo**
J48	0.04 seconds
Naive Bayes	0.01 seconds
SVM	15.19 seconds

4.3 Resultados

O primeiro teste realizado no WEKA com o ficheiro de treino, baseou-se no algoritmo J48 que nos apresenta a árvore de decisão da figura 4.9. Aqui pode-se identificar nos ramos da árvore gerada que avaliações necessárias para identificar o tráfego normal e anómalo.

Neste treino, é possível notar que o nó DPORT assume o valor normal para todos os endereços inferiores ou iguais a 1180. Se for maior, o nó FL faz um controlo assumindo normal todos os valores inferiores ou igual a 20. O nó DPORT é utilizado novamente assumindo normal os valores inferiores ou iguais as 1900. Acima disso é considerado anormal.

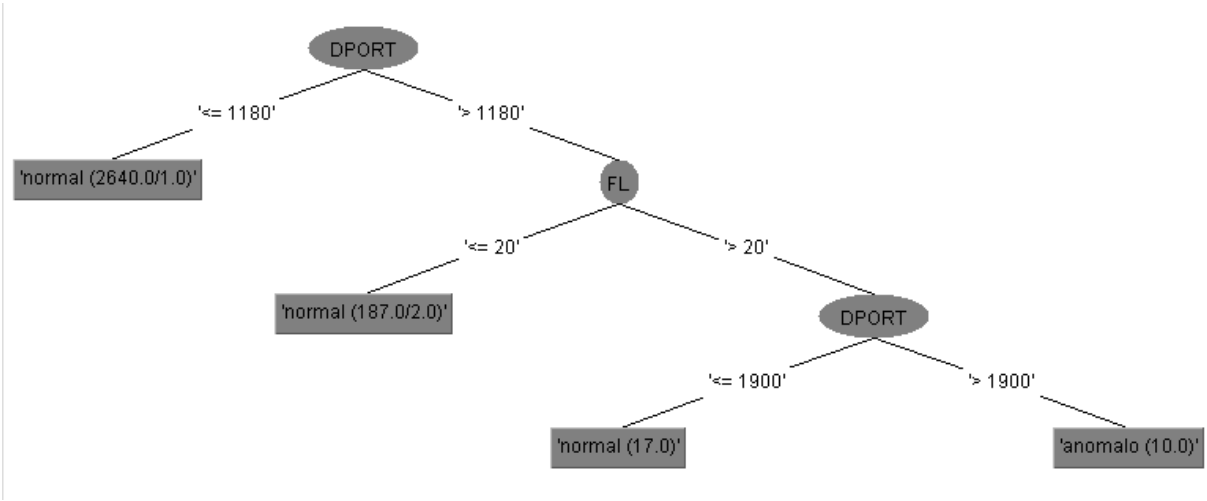


Figura 4.9: Árvore de decisão criada pelo WEKA

Pode-se verificar que foram classificadas corretamente 4277 amostras num percentual de 99.9299% do ficheiro de treino:

```
=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances        4277               99.9299 %
Incorrectly Classified Instances         3                0.0701 %
Kappa statistic                          0.9139
Mean absolute error                      0.0017
Root mean squared error                  0.0264
Relative absolute error                 19.0891 %
Root relative squared error             39.7485 %
Total Number of Instances             4280
```

Verifica-se que, após o treino, na validação do ficheiro de teste que contém 1403 amostras, das quais 164 estão classificadas como trafego anómalo, houve uma classificação correta de 1229 amostras dando um percentual de 87.598% do ficheiro de testes:

```
=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances        1229               87.598  %
```

```
Incorrectly Classified Instances       174               12.402  %
Kappa statistic                         -0.0136
Mean absolute error                      0.124
Root mean squared error                  0.3501
Relative absolute error                102.9558 %
Root relative squared error            102.8777 %
Total Number of Instances             1403
```

Depois de analisar estes valores, optou-se por aumentar o número de amostras do ficheiro de treino que passou a ter 88755 amostras das quais 260 são classificadas como tráfego anómalo. Desta forma os valores obtidos tiveram mudanças significativas a começar pela árvore de decisão apresentada na figura 4.10.

Pode-se verificar no output gerado pelo classificador que o percentual de acerto aumentou para 99.9596% do ficheiro de treino:

```
=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances       88979               99.9596 %
Incorrectly Classified Instances        36                0.0404 %
Kappa statistic                          0.9257
Mean absolute error                      0.0008
Root mean squared error                  0.0201
Relative absolute error                 13.803  %
Root relative squared error             37.1878 %
Total Number of Instances            89015
```

Estes valores foram significativos para o ficheiro de teste após o treino, obtendo agora uma taxa de acerto de 99.4298% com uma diferença de 11.8318% em relação ao primeiro teste:

```
=== Evaluation on test set ===
=== Summary ===

```

```
Correctly Classified Instances        1395               99.4298 %
Incorrectly Classified Instances         8                0.5702 %
Kappa statistic                          0.9718
Mean absolute error                      0.0067
Root mean squared error                  0.0752
Relative absolute error                  5.6215 %
Root relative squared error             22.0589 %
Total Number of Instances             1403
```

Na utilização do algoritmo Naive Bayes os valores conseguidos foram em treino, um acerto de 94.1527% e em teste a taxa de acertos foi 98.3607%:

```
=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances       83810               94.1527 %
Incorrectly Classified Instances      5205                5.8473 %
Kappa statistic                          0.0655
Mean absolute error                      0.0577
Root mean squared error                  0.2286
Relative absolute error                988.1676 %
Root relative squared error            423.5495 %
Total Number of Instances            89015

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances        1380               98.3607 %
Incorrectly Classified Instances        23                1.6393 %
Kappa statistic                          0.9195
Mean absolute error                      0.0194
Root mean squared error                  0.1275
Relative absolute error                 16.2813 %
Root relative squared error             37.4089 %
Total Number of Instances             1403
```

Utiliza-se também um algoritmo SVM que registou valores muito bons apesar de ser o que leva mais tempo a construir o modelo de treino e testes:

```
1  === Evaluation on training set ===
2  === Summary ===
3
4  Correctly Classified Instances       88998               99.9809 %
5  Incorrectly Classified Instances        17                0.0191 %
6  Kappa statistic                          0.9661
7  Mean absolute error                      0.0002
8  Root mean squared error                  0.0138
9  Relative absolute error                  3.2726 %
10 Root relative squared error             25.6078 %
11 Total Number of Instances            89015
12
13 === Evaluation on test set ===
14 === Summary ===
15
16 Correctly Classified Instances        1403              100      %
17 Incorrectly Classified Instances         0                0      %
18 Kappa statistic                          1
19 Mean absolute error                      0
20 Root mean squared error                  0
21 Relative absolute error                  0      %
22 Root relative squared error              0      %
23 Total Number of Instances             1403
```

Para melhor visualizar a diferença entre cada um dos algoritmos apresenta-se uma sintese nas tabelas 4.3 e 4.4.

Tabela 4.3: Treinamento com 89015 amostras

Algorítimo utilizado	**Tempo**	**Classificação correta**
J48	1.96 Segundos	99.96%
Naive Bayes	0.19 Segundos	94.15%
SVM	5219.86 Segundos	99.98%

Tabela 4.4: Teste após treinamento com 1403 amostras

Algorítimo utilizado	**Tempo**	**Classificação correta**
J48	1.88 Segundos	99.43%
Naive Bayes	0.19 Segundos	98.36%
SVM	5134.39 Segundos	100%

Nos anexos A.3, A.4, A.5, A.6, A.7 e A.8, pode-se verificar que a variação nas matrizes de cada um dos algoritmos utilizados e os resultados obtidos em todos os experimentos são praticamente idênticos.

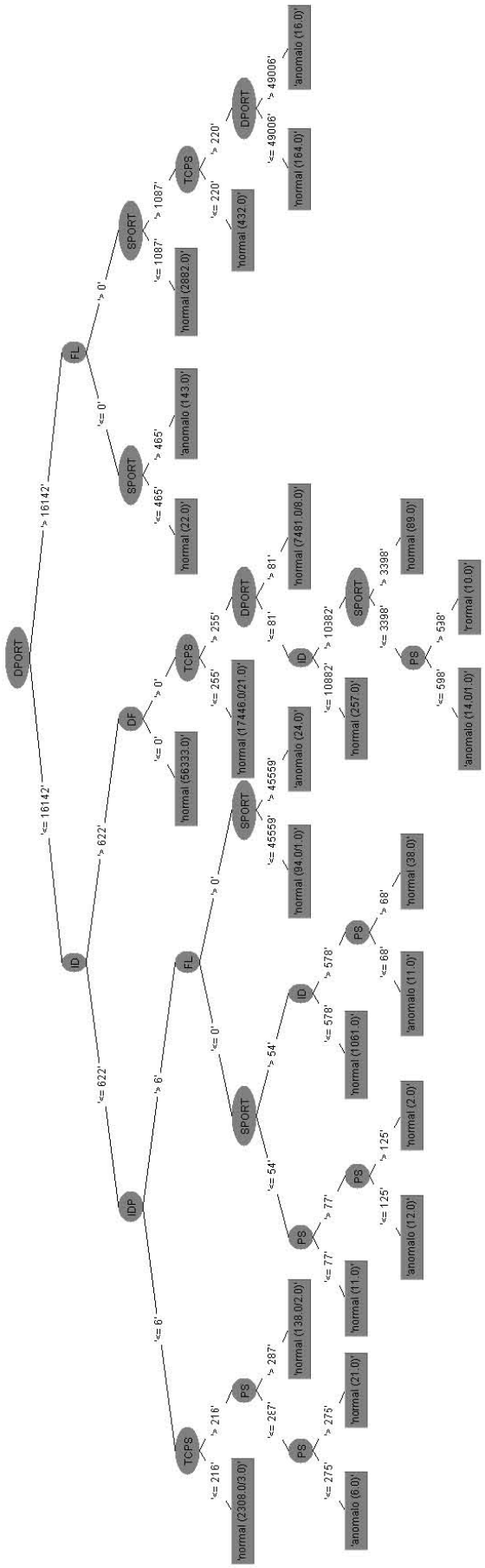


Figura 4.10: Árvore de decisão com novo número de amostras

Capítulo 5

Conclusão e Trabalho Futuro

O estudo sobre os métodos de análise e classificação de *malware* tem o objetivo de criar competências que ajudem a desenvolver regras para a deteção de *malware*. À medida em que se avançava na pesquisa e consulta bibliográfica, apercebemo-nos da fragilidade dos sistemas informáticos perante ameaças de *malware* e da necessidade de estudos mais profundos para criar condições de segurança aceitáveis.

Apesar de existir uma vasta e abrangente quantidade de informação, cada autor utiliza métodos e ferramentas distintas para fazer o mesmo trabalho, tornando a nossa procura muito mais abrangente e consequentemente mais complexa. Em alguns casos os métodos aplicados encontram-se ultrapassados e vivamente aconselhado por vários autores a não utilização pois esses métodos podem gerar falsos positivos. Isso tornou ainda mais difícil chegar a uma conclusão coerente de qual metodologia deveria ser aplicada neste trabalho.

Um dos objetivos passou por construir uma base de dados para a aprendizagem. Considerando que a maior parte dos dados encontrados não são capturados em ambientes reais com IP's não privados, não era possível detetar nem criar modelos fidedignos. Assim, foi criado uma forma automatizada que nos permite capturar tráfego em tempo real dentro de um ambiente virtual controlado, especialmente criado para o efeito, e recorrendo a outro software que utiliza métodos de análise por assinatura (*snort*), foi possível classificar

o tráfego capturado, identificando quais os pacotes que estavam infetados e permitindo extrair as *features* para serem utilizadas na aprendizagem. Também foi desenvolvido uma *script* que permite, com base em tráfego previamente capturado, fazer a classificação. Com isso e o processamento inicial foi possível fazer a aprendizagem. Desta forma foi possível, baseado em assinaturas, criar um modelo que nos permite classificar tráfego não conhecido, respondendo assim a dois objetivos deste trabalho. Não obstante, conseguiu-se aplicar regras simples utilizando os algoritmos J48, Naive Bayes e SVM para classificar, como forma de demonstrar a viabilidade dos dados capturados desta forma.

No decorrer do trabalho experimental algumas opções determinaram o rumo a seguir. Este trabalho representa um ponto de partida para alcançar objetivos com níveis de dificuldade superiores, levando a análise e classificação de *malware* a um nível de compreensão mais avançado, estudando e desenvolvendo ferramentas com capacidades cada vez mais elevadas nesta área.

Assim em relação à captura de informação relativamente ao tráfego de rede, esta deve ser ampliada com utilização de IDS, Honeypots, entre outros, para que se possa criar uma base de dados com informação vital para estudos e análise mais profunda.

Em relação ao modelo utilizado, sugere-se o desenvolvimento de modelos mais detalhados e que tenha um forte teor de exigência acrescido para evitar falsos positivos durante a análise do tráfego de rede.

Por último, sugere-se um estudo para avaliar a potencialidade de aplicações comerciais versus aplicações open source a fim de obter resultados com maior fiabilidade possível.

Bibliografia

Avira Operation (2013). Available from: http://www.avira.com/pt-br/index. Accessed 16/02/2013.

Bellhouse, D. R. (2004). The reverend thomas bayes, FRS: a biography to celebrate the tercentenary of his birth. *Statistical Science*, 19(1):3–43.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.

Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil, C. (2012). *Cartilha de Segurança para Internet*. Available from: http://cartilha.cert.br/livro/, 2ª edition. Accessed 10/09/2013.

Cisco (2011a). Cisco ios netflow version 9 flow-record format. http://www.cisco.com. Accessed 09/05/2013.

Cisco (2011b). Global internet traffic projected to quadruple by 2015. Cisco´s Technology News Site. Avaliable from: http://newsroom.cisco.com/press-release-content?type=webcontent&articleId=324003.

Cohen, F. B. (1986). *Computer Viruses*. PhD thesis, UNIVERSITY OF SOUTHERN CALIFORNIA. Avaliable from: http://all.net/books/Dissertation.pdf.

Cormode, G. and Muthukrishnan, S. (2005). What's new: finding significant differences in network data streams. *IEEE/ACM Trans. Netw.*, 13(6):1219–1232.

Farraposo, S., Owezarski, P., and Monteiro, E. (2007a). Detection, classification et identification d'anomalies de trafic. In: Colloque Francophone d'Ingenierie des Protocoles.

Farraposo, S., Owezarski, P., and Monteiro, E. (2007b). A mult-scale tomographic algorithm for detecting and classifying traffic anomalies. IEEE International Conference.

Filho, D. S. F., Grégio, A. R. A., Afonso, V. M., Santos, R. D. C., Jino, M., and de Geus, P. L. (2010). Análise comportamental de código malicioso através da monitoração de chamadas de sistema e tráfego de rede. *Universidade Estadual de Campinas (Unicamp) - Campinas - SP - Brasil*. Available from: `http://www.las.ic.unicamp.br/paulo/papers/2010-SBSEG-dario.fernandes-andre.gregio-vitor.afonso-rafael.santos-mario.jino-analise.malware.pdf` (accessed september, 10, 2013).

Forouzan, B. (2006). *Comunicacao de Dados E Redes de Computadores*. BOOKMAN COMPANHIA ED.

Franklin, J., Perrig, A., Paxson, V., and Savage, S. (2007). An inquiry into the nature and causes of the wealth of internet miscreants. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 375–388, New York, NY, USA. ACM.

Frederick B. Cohen (1984). Experiments with computer viruses. *UNIVERSITY OF SOUTHERN CALIFORNIA*. Available from: `http://www.all.net/books/virus/part5.html`.

Holz, T., Engelberth, M., and Freiling, F. (2008). Learning more about the underground economy: A case-study of keyloggers and dropzones. *Reihe Informatik TR-2008-006, University of Mannheim*. Available from: `http://honeyblog.org/junkyard/reports/impersonation-attacks-TR.pdf`.

Kruse, W. and Heiser, J. (2002). *Computer forensics: incident response essentials*. Addison-Wesley.

Laboratório da ESET Latinoamérica (2006 (accessed August, 27, 2012)). Cronologia de los virus informáticos. Available from: `http://www.eset-la.com/pdf/prensa/informe/cronologia_virus_informaticos.pdf`.

Lakhina, A., Crovella, M., and Diot, C. (2005). Mining anomalies using traffic feature distributions. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '05, pages 217–228, New York, NY, USA. ACM.

LIMA, C. F. L., ASSIS, F. M., and SOUZA, C. P. (2010). Árvores de decisão baseadas nas entropias de shannon, rényi e tsallis para sistemas tolerantes a intrusão. *La Novena Conferencia Iberoamericana en Sistemas, Cibernética e Informática: CISCI 2010*.

Lucca, G., Pereira, I. A., Prisco, A., and Borges, E. N. ("2013"). Uma implementação do algoritmo naïve bayes para classificação de texto. *Centro de Ciências Computacionais - Universidade Federal do Rio Grande (FURG)*.

Marques, R. L. and Dutra, I. ("2003"). Redes bayesianas: o que são, para que servem, algoritmos e exemplos de aplicações. *Coppe Sistemas – UFRJ*.

Mell, P., Kent, K., Nusbaum, J., of Standards, N. I., and (U.S.), T. (2005). *Guide to Malware incident prevention and handling*. U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology Gaithersburg, MD.

Melo, L., Amaral, D. M., Sakakibara, F., de Almeida, A. R., de Sousa Jr, R. T., and Nascimento, A. (2011). Análise de malware: Investigação de códigos maliciosos através de uma abordagem prática. Available from: `http://dainf.ct.utfpr.edu.br/~maziero/lib/exe/fetch.php/ceseg:2011-sbseg-mc1.pdf`. Accessed 11/09/2013.

M.L.Monsores, A.Ziviani, and P.S.Rodrigues (2012). Detecção de anomalias de tráfego usando entropia não-extensiva. *LNCC/MCT*.

Moore, A. W. and Zuev, D. (2005). Internet traffic classification using bayesian analysis techniques. *SIGMETRICS Perform. Eval. Rev.*, 33(1):50–60.

Moore, D., Keys, K., Koga, R., Lagache, E., and claffy, k. (2001). CoralReef software suite as a tool for system and network administrators. In *Usenix LISA*, pages 4–7, San Diego, CA. Usenix.

MyCERT (2013). Mycert - the malaysian computer emergency response team. Available from: `http://www.mycert.org.my/en/`. Accessed 10/09/2013.

Paxson, V. and Floyd, S. (1995). Wide area traffic: the failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Raphanelli, G. F. (2008). Detecção e classificação de anomalias no tráfego de redes de computadores. Master's thesis, Universidade Federal de Santa Catarina, Forianópolis - SC.

Silva, M., Sampaio, M., Antunes, M., and Frade, M. (2001). Estudo de sistemas de detecção e prevenção de intrusões - uma abordagem open source. *IPL - Escola Superior de Tecnologia e Gestão de Leiria*. Available from: `http://mosel.estg.ipleiria.pt/files/projectos/Paper_-_IDS_%26_Snort_0.pdf`.

Souza, E. P. (2008). Estudo sobre sistema ddetecção de intrusão por anaomalias: uma abordagem utilizando redes neurais. Master's thesis, Universidade Salvador.

Szor, P. (2005). *The Art of Computer Virus Research and Defense*. Addison-Wesley Professional.

Trend Micro Incorporated (2013). Threat encyclopedia. Available from: `http://about-threats.trendmicro.com/us/threatencyclopedia#malware`. Accessed 15/02/2013.

Venere, G. (2009). *Análise forense*. Escola Superior de Redes RNP. Avaliable from: `http://pt.scribd.com/doc/128380450/Analise-Forense`.

Zavala, Y., Stênico, J., and Ling, L. L. (2011 (Accessed 27/09/2013)). Classificação de trafego de internet baseado em cascatas multiplicativas. *Faculdade de Engenharia Elétrica e de Computação - FEEC Universidade Estadual de Campinas (Unicamp).* Avaliable from: `http://www.sps.fee.unicamp.br/sps2011/proceedings_sps2011/Yulios_Multiplicativas_SPS2011.pdf`.

Zhang, H. (2004). The optimality of naive bayes. *American Association for Artificial Intelligence (www.aaai.org).*

Apêndice A

Apêndice Geral

A.1 Código fonte C++

```
1  //============================================================================
2  // Name        : pcap_read.cpp
3  // Author      : Luis Afonso
4  // Version     :
5  // Copyright   : Your copyright notice
6  // Description : Leitura de ficheiros pcap
7  //============================================================================
8
9  #define APP_NAME    "pcap_read"
10 #define APP_DESC    "Leitura de ficheiros pcap"
11 #define APP_COPYRIGHT "Copyright (c) 2013 Luis Afonso"
12 #define APP_DISCLAIMER  "THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM."
13
14 #include <iostream>
15 #include <pcap.h>
16 #include <stdio.h>
17 #include <string.h>
18 #include <stdlib.h>
19 #include <ctype.h>
```

```
20 #include <errno.h>
21 #include <sys/types.h>
22 #include <sys/socket.h>
23 #include <net/ethernet.h>
24 #include <netinet/ip.h>
25 #include <netinet/in.h>
26 #include <netinet/tcp.h>
27 #include <arpa/inet.h>
28 #include <netdb.h>
29 #include <netinet/ip_icmp.h>
30
31 /* default snap length (maximum bytes per packet to capture) */
32 #define SNAP_LEN 1518
33
34 /* ethernet headers are always exactly 14 bytes [1] */
35 #define SIZE_ETHERNET 14
36
37 /* Ethernet addresses are 6 bytes */
38 #define ETHER_ADDR_LEN  6
39
40 /*******   Feature variables *************************/
41 char filename [150];
42 int packet_Count =0, syn_pkt_prob=0 ,  urgent_count=0 ,
43     distinct_window_size_value=0 , distinct_seq_value=0 ,
44     tcp_syn_plus_ack_count=0 , udp_count=0 , icmp_count=0 ,
45     icmp_echo_reply_count=0 , total_bytes_icmp=0 , land_flag=0,
46     fragment_count=0 , invalid_urg_ptr=0 ,tcp_count=0, distinct_TTL_values =0,
47     distinct_source_port=0, distinct_dst_port=0;
48
49 int ack_pkt_prob=0 , fin_pkt_prob=0, urg_ptr_pkt_prob=0 , rst_pkt_prob=0;
50
51 long int    total_bytes_host =0, total_bytes_source =0 ;
52 int first_flag=0;
53 long int window_sizes[10000], sequence_values[10000], source_port_values[65535],
     dst_port_values[65535];
```

```
54 int ttl_values[10000],len;
55 double duration =0.0;
56 double first_ts = 0 , last_ts=0;
57 float tcp_ratio=0.0 , udp_ratio =0.0, icmp_ratio=0.0 ;
58 long   pkt_header_sizes=0 , pkt_data_sizes =0;
59 float pkt_data_avg=0.0, pkt_hdr_avg=0.0;
60 double th = 1.0;
61
62
63
64
65 /* Ethernet header */
66 struct sniff_ethernet {
67         u_char  ether_dhost[ETHER_ADDR_LEN];    /* destination host address */
68         u_char  ether_shost[ETHER_ADDR_LEN];    /* source host address */
69         u_short ether_type;                     /* IP? ARP? RARP? etc */
70 };
71
72 /* IP header */
73 struct sniff_ip {
74         u_char  ip_vhl;                 /* version << 4 | header length >> 2 */
75         u_char  ip_tos;                 /* type of service */
76         u_short ip_len;                 /* total length */
77         u_short ip_id;                  /* identification */
78         u_short ip_off;                 /* fragment offset field */
79         #define IP_RF 0x8000            /* reserved fragment flag */
80         #define IP_DF 0x4000            /* dont fragment flag */
81         #define IP_MF 0x2000            /* more fragments flag */
82         #define IP_OFFMASK 0x1fff       /* mask for fragmenting bits */
83         u_char  ip_ttl;                 /* time to live */
84         u_char  ip_p;                   /* protocol */
85         u_short ip_sum;                 /* checksum */
86         struct  in_addr ip_src,ip_dst;  /* source and dest address */
87 };
88 #define IP_HL(ip)               (((ip)->ip_vhl) & 0x0f)
```

```
89 #define IP_V(ip)                (((ip)->ip_vhl) >> 4)
90
91 /* TCP header */
92 typedef u_int tcp_seq;
93
94 struct sniff_tcp {
95         u_short th_sport;               /* source port */
96         u_short th_dport;               /* destination port */
97         tcp_seq th_seq;                 /* sequence number */
98         tcp_seq th_ack;                 /* acknowledgement number */
99         u_char  th_offx2;               /* data offset, rsvd */
100     #define TH_OFF(th)      (((th)->th_offx2 & 0xf0) >> 4)
101         u_char  th_flags;
102         #define TH_FIN  0x01
103         #define TH_SYN  0x02
104         #define TH_RST  0x04
105         #define TH_PUSH 0x08
106         #define TH_ACK  0x10
107         #define TH_URG  0x20
108         #define TH_ECE  0x40
109         #define TH_CWR  0x80
110         #define TH_FLAGS        (TH_FIN|TH_SYN|TH_RST|TH_ACK|TH_URG|TH_ECE|TH_CWR)
111         u_short th_win;                 /* window */
112         u_short th_sum;                 /* checksum */
113         u_short th_urp;                 /* urgent pointer */
114 };
115
116 struct sniff_udp {
117         u_short udp_sport;               /* source port */
118         u_short udp_dport;               /* destination port */
119         u_short udp_hlen;   /* Udp header length*/
120         u_short udp_chksum;   /* Udp Checksum */
121 };
122
123 struct sniff_icmp
```

```
124 {
125   u_char icmp_type;
126   u_char icmp_code;
127   u_short icmp_checksum;
128 };
129 
130 using namespace std;
131 
132 void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet);
133 void print_hex_ascii_line(const u_char *payload, int len, int offset);
134 void print_payload(const u_char *payload, int len);
135 void print_app_banner(void);
136 void print_app_usage(void);
137 
138 int main(int argc, char *argv[]) {
139   char * dev = NULL;
140   pcap_t *descr;
141   char errbuf[PCAP_ERRBUF_SIZE];
142 
143   char filter_exp[] = "ip";   /* filter expression [3] */
144   struct bpf_program fp;      /* compiled filter program (expression) */
145   // bpf_u_int32 mask;        /* subnet mask */
146   bpf_u_int32 net;        /* ip */
147   //int num_packets = 10;     /* number of packets to capture */
148 
149 
150   print_app_banner();
151 
152   if(argc != 2) {
153 
154     fprintf(stderr, "error:␣unrecognized␣command-line␣options\n\n");
155     print_app_usage();
156     exit(EXIT_FAILURE);
157 
158   } else {
```

```
159     // convert argument to file name
160     dev = argv[1];
161
162     // get the file name
163     string file = dev;
164
165     // open capture file for offline processing
166     descr = pcap_open_offline(file.c_str(), errbuf);
167     if (descr == NULL) {
168       cout << "pcap_open_offline() failed: " << errbuf << endl;
169       return 1;
170     }
171
172   }
173
174
175   /* print capture info */
176   printf("File: %s\n", dev);
177
178   /* compile the filter expression */
179   if (pcap_compile(descr, &fp, filter_exp, 0, net) == -1) {
180     fprintf(stderr, "Couldn't parse filter %s: %s\n",
181       filter_exp, pcap_geterr(descr));
182     exit(EXIT_FAILURE);
183   }
184
185   /* apply the compiled filter */
186   if (pcap_setfilter(descr, &fp) == -1) {
187     fprintf(stderr, "Couldn't install filter %s: %s\n",
188       filter_exp, pcap_geterr(descr));
189     exit(EXIT_FAILURE);
190   }
191
192   // start packet processing loop, just like live capture
193   if (pcap_loop(descr, 0, got_packet, NULL) < 0) {
```

```
194     cout << "pcap_loop()␣failed:␣" << pcap_geterr(descr);
195     return 1;
196   }
197
198
199
200   /* cleanup */
201   pcap_freecode(&fp);
202   pcap_close(descr);
203
204
205   cout << "capture␣finished" << endl;
206
207   return 0;
208 }
209
210
211
212
213 /*
214  * print data in rows of 16 bytes: offset   hex   ascii
215  *
216  * 00000   47 45 54 20 2f 20 48 54  54 50 2f 31 2e 31 0d 0a   GET / HTTP/1.1..
217  */
218 void print_hex_ascii_line(const u_char *payload, int len, int offset)
219 {
220
221   int i;
222   int gap;
223   const u_char *ch;
224
225   /* offset */
226   printf("%05d␣␣␣", offset);
227
228   /* hex */
```

```
229   ch = payload;
230   for(i = 0; i < len; i++) {
231     printf("%02x␣", *ch);
232     ch++;
233     /* print extra space after 8th byte for visual aid */
234     if (i == 7)
235       printf("␣");
236   }
237   /* print space to handle line less than 8 bytes */
238   if (len < 8)
239     printf("␣");
240
241   /* fill hex gap with spaces if not full line */
242   if (len < 16) {
243     gap = 16 - len;
244     for (i = 0; i < gap; i++) {
245       printf("␣␣␣");
246     }
247   }
248   printf("␣␣␣");
249
250   /* ascii (if printable) */
251   ch = payload;
252   for(i = 0; i < len; i++) {
253     if (isprint(*ch))
254       printf("%c", *ch);
255     else
256       printf(".");
257     ch++;
258   }
259
260   printf("\n");
261
262 return;
263 }
```

```
264
265 /*
266  * print packet payload data (avoid printing binary data)
267  */
268 void print_payload(const u_char *payload, int len)
269 {
270
271   int len_rem = len;
272   int line_width = 16;      /* number of bytes per line */
273   int line_len;
274   int offset = 0;         /* zero-based offset counter */
275   const u_char *ch = payload;
276
277   if (len <= 0)
278     return;
279
280   /* data fits on one line */
281   if (len <= line_width) {
282     print_hex_ascii_line(ch, len, offset);
283     return;
284   }
285
286   /* data spans multiple lines */
287   for ( ;; ) {
288     /* compute current line length */
289     line_len = line_width % len_rem;
290     /* print line */
291     print_hex_ascii_line(ch, line_len, offset);
292     /* compute total remaining */
293     len_rem = len_rem - line_len;
294     /* shift pointer to remaining bytes to print */
295     ch = ch + line_len;
296     /* add offset */
297     offset = offset + line_width;
298     /* check if we have line width chars or less */
```

```
299     if (len_rem <= line_width) {
300       /* print last line and get out */
301       print_hex_ascii_line(ch, len_rem, offset);
302       break;
303     }
304   }
305
306 return;
307 }
308
309 /*
310  * dissect/print packet
311  */
312 void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet)
313 {
314
315   static int count = 1;                   /* packet counter */
316
317   /* declare pointers to packet headers */
318   const struct sniff_ethernet *ethernet;  /* The ethernet header [1] */
319   const struct sniff_ip *ip;              /* The IP header */
320   const struct sniff_tcp *tcp;            /* The TCP header */
321   const u_char *payload;                    /* Packet payload */
322
323   struct sniff_udp *udp;            /* The Udp header */
324   struct sniff_icmp *icmp;      /* The ICMP header*/
325
326
327   int size_ip;
328   int size_tcp;
329   int size_payload;
330   int size_udp;
331
332
333   printf("\nPacket␣number␣%d:", count);
```

```
334   count++;
335   printf("\n␣timestamp␣seconds.......:␣%ld", (long int)header->ts.tv_sec);
336   printf("\n␣timestamp␣micro␣seconds.:␣%ld", (long int)header->ts.tv_usec);
337
338   /* define ethernet header */
339   ethernet = (struct sniff_ethernet*)(packet);
340
341   /* define/compute ip header offset */
342   ip = (struct sniff_ip*)(packet + SIZE_ETHERNET);
343   size_ip = IP_HL(ip)*4;
344   if (size_ip < 20) {
345     printf("␣␣␣*␣Invalid␣IP␣header␣length:␣%u␣bytes\n", size_ip);
346     return;
347   }
348   printf("\n\n␣successfully␣casted␣to␣ip␣pcaket\n\n");
349
350   /* print source and destination IP addresses */
351   printf("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣From.:␣%s\n", inet_ntoa(ip->ip_src));
352   printf("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣To.:␣%s\n", inet_ntoa(ip->ip_dst));
353
354   if((ip->ip_src.s_addr== ip->ip_dst.s_addr)) {
355     land_flag = 1;
356   }
357   printf("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣more␣flag.:␣%d\n",ip->ip_off);
358   if(ip->ip_off > 1)  {
359     fragment_count++;
360   }
361
362   /*  pkt_data_sizes calculation */
363   pkt_data_sizes += ip->ip_len - (ip->ip_vhl>>2);
364   pkt_header_sizes+= (ip->ip_vhl>>2);
365
366   int i;
367
368   for(i=0;i<distinct_TTL_values;i++)  {
```

```
369     if(ttl_values[i]== ip->ip_ttl)
370       break;
371     else
372       continue;
373   }
374   if(i == distinct_TTL_values) {
375     ttl_values[i]= ip->ip_ttl;
376     distinct_TTL_values++;
377   }
378   len     = ntohs(ip->ip_len);
379   //int length -= sizeof(struct ether_header);
380
381   printf("\n␣␣␣␣␣␣␣␣␣␣␣␣␣␣packet␣length:␣%d\n", len);
382   packet_Count ++;
383
384   /* timestamp calculation */
385   double ts = header->ts.tv_sec + ((double) header->ts.tv_usec) / 1000000;
386   if (first_flag == 0) {
387     first_ts = ts;
388     first_flag =1;
389   }
390   //if (ts < last_ts)
391
392   last_ts = ts;
393   if((last_ts - first_ts)> th) {
394     //printtofile();
395     //initialize();
396     first_flag =0;
397   }
398
399   /* determine protocol */
400   switch(ip->ip_p) {
401     case IPPROTO_TCP:
402       printf("␣␣␣Protocol:␣TCP\n");
403       /* define/compute tcp header offset */
```

```
404       tcp = (struct sniff_tcp*)(packet + SIZE_ETHERNET + size_ip);
405       size_tcp = TH_OFF(tcp)*4;
406       if (size_tcp < 20) {
407         printf("␣␣␣*␣Invalid␣TCP␣header␣length:␣%u␣bytes\n", size_tcp);
408         return;
409       }
410       pkt_data_sizes-=size_tcp;
411       pkt_header_sizes+= size_tcp;
412       tcp_count++;
413       //FILE  *qp = fopen("features.txt","a");
414       if(((tcp->th_flags)&TH_SYN) == 2)
415       {
416         syn_pkt_prob++;
417       }
418
419       if(((tcp->th_flags)&TH_SYN) == 2 &&((tcp->th_flags)&TH_ACK) == 16 )
420       {
421         tcp_syn_plus_ack_count ++;
422       }
423           // printf("\n number of syn packets are: %d",syn_pkt_prob++);
424         //fclose(qp);
425
426       if(((tcp->th_flags)&TH_FIN) == 1)
427       {
428         fin_pkt_prob++;
429       }
430
431       if(((tcp->th_flags)&TH_ACK) == 16)
432       {
433         ack_pkt_prob++;
434       }
435
436       if(((tcp->th_flags)&TH_URG) == 64)
437       {
438         urg_ptr_pkt_prob++;
```

```
439       }
440
441       if(((tcp->th_flags)&TH_RST) == 4)
442       {
443         rst_pkt_prob++;
444       }
445
446       //int i;
447       for(i=0;i<distinct_window_size_value;i++)
448       {
449         if(window_sizes[i]==tcp->th_win)
450           break;
451         else
452           continue;
453       }
454
455       if(i== distinct_window_size_value)
456       {
457         window_sizes[i]=tcp->th_win;
458         distinct_window_size_value++;
459       }
460
461       printf("packet␣number␣%d␣is␣recived",packet_Count);
462           //fprintf(qp,"\ndistinct window size values :%u",tcp->th_win);
463
464       for(i=0;i<distinct_seq_value;i++)
465       {
466         if(sequence_values[i]==tcp->th_seq)
467           break;
468         else
469           continue;
470       }
471
472       if(i== distinct_seq_value)
473       {
```

```
474         sequence_values[i]=tcp->th_seq;
475         distinct_seq_value++;
476       }
477
478       for(i=0;i<distinct_source_port;i++)
479       {
480         if(source_port_values[i]==tcp->th_sport)
481           break;
482         else
483           continue;
484       }
485
486       if(i== distinct_source_port)
487       {
488         source_port_values[i]=tcp->th_sport;
489         distinct_source_port++;
490       }
491
492       for(i=0;i<distinct_dst_port;i++)
493       {
494         if(dst_port_values[i]==tcp->th_dport)
495           break;
496         else
497           continue;
498       }
499
500       if(i== distinct_dst_port)
501       {
502         dst_port_values[i]=tcp->th_dport;
503         distinct_dst_port++;
504       }
505
506       printf("\n");
507       printf("␣␣␣Src␣port:␣%d\n", ntohs(tcp->th_sport));
508       printf("␣␣␣Dst␣port:␣%d\n", ntohs(tcp->th_dport));
```

```
509
510       /* define/compute tcp payload (segment) offset */
511       payload = (u_char *)(packet + SIZE_ETHERNET + size_ip + size_tcp);
512
513       /* compute tcp payload (segment) size */
514       size_payload = ntohs(ip->ip_len) - (size_ip + size_tcp);
515
516       //fclose(qp);
517       //return;
518       break;
519     case IPPROTO_UDP:
520       printf("   Protocol: UDP\n");
521       udp = (struct sniff_udp*)(packet + SIZE_ETHERNET + size_ip);
522       /*size_udp = TH_OFF(udp)*4;
523       if (size_udp < 8) {
524         printf("   * Invalid UDP header length: %u0 bytes\n", size_udp);
525         return;
526       }*/
527       pkt_data_sizes-=size_udp;
528       pkt_header_sizes+= size_udp;
529       udp_count++;
530       for(i=0;i<distinct_source_port;i++) {
531         if(source_port_values[i]==udp->udp_sport)
532           break;
533         else
534           continue;
535       }
536       if(i== distinct_source_port) {
537         source_port_values[i]=udp->udp_sport;
538         distinct_source_port++;
539       }
540       for(i=0;i<distinct_dst_port;i++) {
541         if(dst_port_values[i]==udp->udp_dport)
542           break;
543         else
```

```
544             continue;
545         }
546         if(i== distinct_dst_port) {
547           dst_port_values[i]=udp->udp_dport;
548           distinct_dst_port++;
549         }
550 
551 
552         printf("\n");
553         printf("␣␣␣Src␣port:␣%d\n", ntohs(udp->udp_sport));
554         printf("␣␣␣Dst␣port:␣%d\n", ntohs(udp->udp_dport));
555 
556         /* define/compute tcp payload (segment) offset */
557         payload = (u_char *)(packet + SIZE_ETHERNET + size_ip + size_udp);
558 
559         /* compute tcp payload (segment) size */
560         size_payload = ntohs(ip->ip_len) - (size_ip + size_udp);
561 
562         break;
563 
564       case IPPROTO_ICMP:
565         printf("␣␣␣Protocol:␣ICMP\n");
566         printf("packet␣number␣%d␣is␣recived",packet_Count);
567         icmp_count++;
568         pkt_data_sizes-= 8;
569         pkt_header_sizes+= 8;
570         icmp = (struct sniff_icmp*)(packet + SIZE_ETHERNET + size_ip);
571         printf("icmp␣type␣value␣is␣%d",icmp->icmp_type);
572         if(icmp->icmp_type == 0)
573             icmp_echo_reply_count++;
574         printf("icmp␣code␣value␣is␣%d",icmp->icmp_code);
575         return;
576 
577       case IPPROTO_IP:
578         printf("␣␣␣Protocol:␣IP\n");
```

```
579       printf("packet␣number␣%d␣is␣recived",packet_Count);
580       return;
581     case IPPROTO_FRAGMENT:
582       printf("␣␣␣Protocol:␣IPv6-Frag\n");
583       return;
584     case IPPROTO_RSVP:
585         printf("␣␣␣Protocol:␣RSVP\n");
586         return;
587     case IPPROTO_ENCAP:
588         printf("␣␣␣Protocol:␣IP-in-IP\n");
589         return;
590     case IPPROTO_AH:
591         printf("␣␣␣Protocol:␣AH\n");
592         return;
593     case IPPROTO_ESP:
594         printf("␣␣␣Protocol:␣ESP\n");
595         return;
596     case IPPROTO_ICMPV6:
597         printf("␣␣␣Protocol:␣ICMPv6\n");
598         return;
599     case IPPROTO_DSTOPTS:
600         printf("␣␣␣Protocol:␣IPv6-DstOpts\n");
601         return;
602     case IPPROTO_IGMP:
603         printf("␣␣␣Protocol:␣IGMP\n");
604         return;
605     case IPPROTO_GGP:
606         printf("␣␣␣Protocol:␣GGP\n");
607         return;
608     case IPPROTO_EGP:
609         printf("␣␣␣Protocol:␣EGP\n");
610         return;
611     case IPPROTO_PUP:
612         printf("␣␣␣Protocol:␣PUP\n");
613         return;
```

```
614     case IPPROTO_IDP:
615         printf("␣␣␣Protocol:␣IDP\n");
616         return;
617     case IPPROTO_HELLO:
618         printf("␣␣␣Protocol:␣HELLO\n");
619         return;
620     case IPPROTO_ND:
621         printf("␣␣␣Protocol:␣ND\n");
622         return;
623     case IPPROTO_EON:
624         printf("␣␣␣Protocol:␣EON\n");
625         return;
626     case IPPROTO_RAW:
627         printf("␣␣␣Protocol:␣RAW\n");
628         return;
629 
630     default:
631       printf("␣␣␣Protocol:␣unknown\n");
632       return;
633   }
634 
635 
636 
637   //printf("\n");
638   //printf("   Src port: %d\n", ntohs(tcp->th_sport));
639   //printf("   Dst port: %d\n", ntohs(tcp->th_dport));
640 
641   /* define/compute tcp payload (segment) offset */
642   //payload = (u_char *)(packet + SIZE_ETHERNET + size_ip + size_tcp);
643 
644   /* compute tcp payload (segment) size */
645   //size_payload = ntohs(ip->ip_len) - (size_ip + size_tcp);
646 
647   /*
648    * Print payload data; it might be binary, so don't just
```

```
649     * treat it as a string.
650     */
651    if (size_payload > 0) {
652      printf("␣␣␣Payload␣(%d␣bytes):\n", size_payload);
653      print_payload(payload, size_payload);
654    }
655
656 return;
657 }
658
659
660
661 /*
662  * app name/banner
663  */
664 void print_app_banner(void)
665 {
666
667    printf("%s␣-␣%s\n", APP_NAME, APP_DESC);
668    printf("%s\n", APP_COPYRIGHT);
669    printf("%s\n", APP_DISCLAIMER);
670    printf("\n");
671
672 return;
673 }
674
675 /*
676  * print help text
677  */
678 void print_app_usage(void)
679 {
680
681    printf("Usage:␣%s␣[filenape.pcap]\n", APP_NAME);
682    printf("\n");
683    printf("Options:\n");
```

```
684   printf("␣␣␣␣␣filename.pcap␣␣␣␣␣Read␣on␣<filename.pcap>␣for␣packets.\n");
685   printf("\n");
686
687 return;
688 }
```

A.2 Código fonte Ruby

```
1  #!/usr/bin/env ruby
2  require 'rubygems'
3  require 'pcap'
4  require 'pcaplet'
5  require 'csv'
6  require 'bson'
7  require 'optparse'
8
9
10 OptionParser.new do |o|
11   o.on('-d') { |b| $quiet = b }
12   o.on('-t') { |b| $teste = b }
13   o.on('-f␣FILENAME') { |filename| $filename = filename }
14   o.on('-o␣OUTPUT') { |output| $output = output }
15   o.on('-h') { puts o; exit }
16   o.parse!
17 end
18
19 unless $filename
20    puts 'Usuage:'
21    puts '␣␣␣␣␣ruby␣read_pcap3c.rb␣-f␣FILEIN␣[-o␣FILEOUT]␣'
22    exit
23 end
24
25 unless $output
```

```
26      unless $teste
27        $output = 'outcsv.csv'
28    else
29        $output = 'outteste.csv'
30      end
31  end
32
33  class Time
34    # tcpdump style format
35    def to_s
36      sprintf "%0.2d:%0.2d:%0.2d.%0.6d", hour, min, sec, tv_usec
37    end
38  end
39
40  def line_containing(file, str)
41    if open(file).grep(/#{str}/).length > 0
42       #match
43       return true
44    else
45       #no match
46       return false
47    end
48  end
49
50
51  # file? will only return true for files
52  if File.file?($filename)
53      inFile = Pcap::Capture.open_offline($filename)
54  else
55      puts "ficheiro: #{$filename} nao foi encontrado."
56      abort
57  end
58
59  # inFile.setfilter("ip")
60  packet_count = 0
```

```
61
62 #HTTP_REQUEST  = Pcap::Filter.new('tcp and dst port 80', inFile)
63 #HTTP_RESPONSE = Pcap::Filter.new('tcp and src port 80', inFile)
64
65 CSV.open($output,"w") do |csv|
66
67     csv << ["No.",
68             "TIME",
69             "EFT",
70             "HS",
71             "TOS",
72             "PS",
73             "ID",
74             "FL",
75             "FR",
76             "PT",
77             "SIP",
78             "SPORT",
79             "DIP",
80             "DPORT",
81             "UDPS",
82             "TCPS",
83             "TCPHS",
84             "IDP",
85             "ACK",
86             "WIN",
87             "DF",
88             "TP"]
89
90     file = "alert"
91     conta = 0
92
93     inFile.loop(-1) do |pkt|
94
95         # Variaveis de parser
```

```
96
97          eft = 0           #Ethernet Frame Type
98          hs = 0            #Header Size
99      tos = 0           #Type of Service
100     ps = 0            #Packet Size
101     id = 0            #Identificador
102     fl = 0            #flags
103     fr = 0            #Fragment
104     pt = "NULL"       #protocolo
105     sip = "NULL"      #source ip
106     sport = 0         #Source Port
107     dip = "NULL"      #dest ip
108     dport = 0         #Destination Port
109     udps  = 0         #UDP Size
110     tcps  = 0         #TCP Size
111     tcphs = 0         #TCP Head Size
112     tp = "?"           #Tipo (Normal, anomalo, ?)
113       idp =0            #Identificacao do protocolo
114         ack = 0
115         win = "NULL"
116         df  = 0
117         data = "NULL"
118   s = "NULL"
119
120
121     #Process packet.
122     packet_count +=  1
123     eft = pkt.datalink              #Ethernet Frame Type
124
125
126
127     ps = pkt.caplen       #Packet Size
128
129
130   if pkt.ip?
```

```
131         #id = pkt.ip_id         #Identificador
132         #sip = pkt.src          #Source IP
133         #dip = pkt.dst          #Dest  IP
134         pt = "IP"              #protocolo
135         sport = 0              #Source Port
136         dport = 0              #Destination Port
137         #tos = pkt.ip_tos       #Type of Service
138         fl = pkt.ip_flags      #flags
139                 df = 1 if pkt.ip_df?
140     end
141     if pkt.tcp?
142         pt = "TCP"
143         tcps  = pkt.tcp_data_len    #TCP Size
144         sport = pkt.sport           #Source Port
145         dport = pkt.dport           #Destination Port
146         tcphs = pkt.tcp_hlen   #TCP Head Size
147         fl = pkt.tcp_flags      #flags
148         fr = pkt.tcp_seq      #Fragment
149         ack = pkt.tcp_ack  if pkt.tcp_ack?
150         win = pkt.tcp_win
151         data = pkt.tcp_data
152         if data and data =~ /^GET\s+(\S+)/
153             path = $1
154             host = pkt.dst.to_s
155             host << ":#{pkt.dport}" #if pkt.dport != 80
156             s = "---->␣#{pkt.src}:#{pkt.sport}␣>␣GET␣http://#{host}#{path}"
157             #s << data
158         end
159     end
160     if pkt.udp?
161         pt = "UDP"
162         udps  = pkt.udp_len    #UDP Size
163         sport = pkt.sport      #Source Port
164         dport = pkt.dport      #Destination Port
165     end
```

```
166
167
168     if pt != "NULL"
169     idp  =  pkt.ip_proto
170     sip  =  pkt.src          #Source IP
171     dip  =  pkt.dst          #Dest  IP
172     hs   =  pkt.ip_hlen      #Header Size
173     id   =  pkt.ip_id        #Identificador
174     tos  =  pkt.ip_tos       #Type of Service
175
176     # Procura no ficheiro snort/alert por aviso
177     str = "#{pkt.time}"
178     unless $teste
179       if line_containing(file,str[2, 13])
180         conta = conta + 1
181           tp="anomalo"
182         else
183           tp="normal"
184         end
185       else
186         tp ="?"
187       end
188           csv << ["#{packet_count}",
189             "#{pkt.time}",
190             "#{eft}",
191             "#{hs}",
192             "#{tos}",
193             "#{ps}",
194             "#{id}",
195             "#{fl}",
196             "#{fr}",
197             "#{pt}",
198             "#{sip}",
199             "#{sport}",
200             "#{dip}",
```

```
201             "#{dport}",
202             "#{udps}",
203             "#{tcps}",
204             "#{tcphs}",
205         "#{idp}",
206             "#{ack}",
207             "#{win}",
208             "#{df}",
209             "#{tp}"]
210       end
211   end
212     puts "foram␣encontradas␣#{conta}␣ocorrencias." unless $teste
213 end
```

A.3 Log de aprendizagem algoritmo J48

```
1  === Run information ===
2
3  Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
4  Relation:     outcsv
5  Instances:    89015
6  Attributes:   17
7                EFT
8                HS
9                TOS
10               PS
11               ID
12               FL
13               FR
14               PT
15               SPORT
16               DPORT
17               UDPS
```

```
18              TCPS
19              TCPHS
20              IDP
21              ACK
22              DF
23              TP
24 Test mode:evaluate on training data
25
26 === Classifier model (full training set) ===
27
28 J48 pruned tree
29 ------------------
30
31 DPORT <= 16142
32 |   ID <= 622
33 |   |   IDP <= 6
34 |   |   |   TCPS <= 216: normal (2308.0/3.0)
35 |   |   |   TCPS > 216
36 |   |   |   |   PS <= 287
37 |   |   |   |   |   PS <= 275: anomalo (6.0)
38 |   |   |   |   |   PS > 275: normal (21.0)
39 |   |   |   |   PS > 287: normal (138.0/2.0)
40 |   |   IDP > 6
41 |   |   |   FL <= 0
42 |   |   |   |   SPORT <= 54
43 |   |   |   |   |   PS <= 77: normal (11.0)
44 |   |   |   |   |   PS > 77
45 |   |   |   |   |   |   PS <= 125: anomalo (12.0)
46 |   |   |   |   |   |   PS > 125: normal (2.0)
47 |   |   |   |   SPORT > 54
48 |   |   |   |   |   ID <= 578: normal (1061.0)
49 |   |   |   |   |   ID > 578
50 |   |   |   |   |   |   PS <= 68: anomalo (11.0)
51 |   |   |   |   |   |   PS > 68: normal (38.0)
52 |   |   |   FL > 0
```

```
53 |   |   |   |   SPORT <= 45559: normal (94.0/1.0)
54 |   |   |   |   SPORT > 45559: anomalo (24.0)
55 |   ID > 622
56 |   |   DF <= 0: normal (56333.0)
57 |   |   DF > 0
58 |   |   |   TCPS <= 255: normal (17446.0/21.0)
59 |   |   |   TCPS > 255
60 |   |   |   |   DPORT <= 81
61 |   |   |   |   |   ID <= 10882: normal (257.0)
62 |   |   |   |   |   ID > 10882
63 |   |   |   |   |   |   SPORT <= 3398
64 |   |   |   |   |   |   |   PS <= 598: anomalo (14.0/1.0)
65 |   |   |   |   |   |   |   PS > 598: normal (10.0)
66 |   |   |   |   |   |   SPORT > 3398: normal (89.0)
67 |   |   |   |   DPORT > 81: normal (7481.0/8.0)
68 DPORT > 16142
69 |   FL <= 0
70 |   |   SPORT <= 465: normal (22.0)
71 |   |   SPORT > 465: anomalo (143.0)
72 |   FL > 0
73 |   |   SPORT <= 1087: normal (2882.0)
74 |   |   SPORT > 1087
75 |   |   |   TCPS <= 220: normal (432.0)
76 |   |   |   TCPS > 220
77 |   |   |   |   DPORT <= 49006: normal (164.0)
78 |   |   |   |   DPORT > 49006: anomalo (16.0)
79
80 Number of Leaves  :   25
81
82 Size of the tree :  49
83
84
85 Time taken to build model: 1.96 seconds
86
87 === Evaluation on training set ===
```

```
=== Summary ===

Correctly Classified Instances       88979               99.9596 %
Incorrectly Classified Instances        36                0.0404 %
Kappa statistic                          0.9257
Mean absolute error                      0.0008
Root mean squared error                  0.0201
Relative absolute error                 13.803  %
Root relative squared error             37.1878 %
Total Number of Instances            89015

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.135      1         1         1          0.981    normal
                 0.865     0          0.996     0.865     0.926      0.981    anomalo
Weighted Avg.    1         0.134      1         1         1          0.981

=== Confusion Matrix ===

     a     b   <-- classified as
 88754     1 |     a = normal
    35   225 |     b = anomalo
```

A.4 Log de Teste com algoritmo J48

```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     outcsv
Instances:    89015
Attributes:   17
              EFT
```

```
8               HS
9               TOS
10              PS
11              ID
12              FL
13              FR
14              PT
15              SPORT
16              DPORT
17              UDPS
18              TCPS
19              TCPHS
20              IDP
21              ACK
22              DF
23              TP
24 Test mode:user supplied test set: size unknown (reading incrementally)
25
26 === Classifier model (full training set) ===
27
28 J48 pruned tree
29 ------------------
30
31 DPORT <= 16142
32 |   ID <= 622
33 |   |   IDP <= 6
34 |   |   |   TCPS <= 216: normal (2308.0/3.0)
35 |   |   |   TCPS > 216
36 |   |   |   |   PS <= 287
37 |   |   |   |   |   PS <= 275: anomalo (6.0)
38 |   |   |   |   |   PS > 275: normal (21.0)
39 |   |   |   |   PS > 287: normal (138.0/2.0)
40 |   |   IDP > 6
41 |   |   |   FL <= 0
42 |   |   |   |   SPORT <= 54
```

```
|   |   |   |   |   PS <= 77: normal (11.0)
|   |   |   |   |   PS > 77
|   |   |   |   |   |   PS <= 125: anomalo (12.0)
|   |   |   |   |   |   PS > 125: normal (2.0)
|   |   |   |   SPORT > 54
|   |   |   |   |   ID <= 578: normal (1061.0)
|   |   |   |   |   ID > 578
|   |   |   |   |   |   PS <= 68: anomalo (11.0)
|   |   |   |   |   |   PS > 68: normal (38.0)
|   |   |   FL > 0
|   |   |   |   SPORT <= 45559: normal (94.0/1.0)
|   |   |   |   SPORT > 45559: anomalo (24.0)
|   ID > 622
|   |   DF <= 0: normal (56333.0)
|   |   DF > 0
|   |   |   TCPS <= 255: normal (17446.0/21.0)
|   |   |   TCPS > 255
|   |   |   |   DPORT <= 81
|   |   |   |   |   ID <= 10882: normal (257.0)
|   |   |   |   |   ID > 10882
|   |   |   |   |   |   SPORT <= 3398
|   |   |   |   |   |   |   PS <= 598: anomalo (14.0/1.0)
|   |   |   |   |   |   |   PS > 598: normal (10.0)
|   |   |   |   |   |   SPORT > 3398: normal (89.0)
|   |   |   |   DPORT > 81: normal (7481.0/8.0)
DPORT > 16142
|   FL <= 0
|   |   SPORT <= 465: normal (22.0)
|   |   SPORT > 465: anomalo (143.0)
|   FL > 0
|   |   SPORT <= 1087: normal (2882.0)
|   |   SPORT > 1087
|   |   |   TCPS <= 220: normal (432.0)
|   |   |   TCPS > 220
|   |   |   |   DPORT <= 49006: normal (164.0)
```

```
78 |   |   |   |   DPORT > 49006: anomalo (16.0)
79
80 Number of Leaves  :   25
81
82 Size of the tree :  49
83
84
85 Time taken to build model: 1.88 seconds
86
87 === Predictions on test split ===
88
89 inst#,    actual, predicted, error, probability distribution (HS)
90      1   1:normal   1:normal         *1      0     (5)
91      2   1:normal   1:normal         *1      0     (5)
92      3   1:normal   1:normal         *1      0     (5)
93      4   1:normal   1:normal         *1      0     (5)
94      5   1:normal   1:normal         *1      0     (5)
95      6   1:normal   1:normal         *0.989  0.011 (5)
96      7   1:normal   1:normal         *0.999  0.001 (6)
97      8   1:normal   1:normal         *0.999  0.001 (6)
98      9   1:normal   1:normal         *1      0     (5)
99     10   1:normal   1:normal         *1      0     (5)
100    11   1:normal   1:normal         *0.999  0.001 (5)
101    12   1:normal   1:normal         *0.999  0.001 (5)
102    13   1:normal   1:normal         *0.999  0.001 (5)
103    14   1:normal   1:normal         *0.986  0.014 (5)
104    15   1:normal   1:normal         *0.999  0.001 (5)
105    16   1:normal   1:normal         *0.999  0.001 (5)
106    17   1:normal   1:normal         *0.999  0.001 (5)
107    18   1:normal   1:normal         *0.999  0.001 (5)
108    19   1:normal   1:normal         *0.999  0.001 (5)
109    20   1:normal   1:normal         *0.999  0.001 (5)
110    21   1:normal   1:normal         *0.999  0.001 (5)
111    22   1:normal   1:normal         *0.999  0.001 (5)
112    23   1:normal   1:normal         *0.999  0.001 (5)
```

```
113     24   1:normal   1:normal         *0.999  0.001 (5)
114     25   1:normal   1:normal         *0.999  0.001 (5)
115     26   1:normal   1:normal         *1      0     (5)
116     27   1:normal   1:normal         *1      0     (5)
117     28   1:normal   1:normal         *0.999  0.001 (5)
118     29   1:normal   1:normal         *0.999  0.001 (5)
119     30   1:normal   1:normal         *0.999  0.001 (5)
120     31   1:normal   1:normal         *0.999  0.001 (5)
121     32   1:normal   1:normal         *0.999  0.001 (5)
122     33   1:normal   1:normal         *0.999  0.001 (5)
123     34   1:normal   1:normal         *0.999  0.001 (5)
124     35   1:normal   1:normal         *0.999  0.001 (5)
125     36   1:normal   1:normal         *0.999  0.001 (5)
126     37   1:normal   1:normal         *0.986  0.014 (5)
127     38   1:normal   1:normal         *0.986  0.014 (5)
128     39   1:normal   1:normal         *0.986  0.014 (5)
129     40   1:normal   1:normal         *0.999  0.001 (5)
130     41   1:normal   1:normal         *0.999  0.001 (5)
131     42   1:normal   1:normal         *0.999  0.001 (5)
132     43   1:normal   1:normal         *0.999  0.001 (5)
133     44   1:normal   1:normal         *0.999  0.001 (5)
134     45   1:normal   1:normal         *0.999  0.001 (5)
135     46   1:normal   1:normal         *0.999  0.001 (5)
136     47   1:normal   1:normal         *0.999  0.001 (5)
137     48   1:normal   1:normal         *0.999  0.001 (5)
138     49   1:normal   1:normal         *0.999  0.001 (5)
139     50   1:normal   1:normal         *0.999  0.001 (5)
140     51   1:normal   1:normal         *0.999  0.001 (5)
141     52   1:normal   1:normal         *0.999  0.001 (5)
142     53   1:normal   1:normal         *0.999  0.001 (5)
143     54   1:normal   1:normal         *0.999  0.001 (5)
144     55   1:normal   1:normal         *0.999  0.001 (5)
145     56   1:normal   1:normal         *0.999  0.001 (5)
146     57   1:normal   1:normal         *0.999  0.001 (5)
147     58   1:normal   1:normal         *0.999  0.001 (5)
```

```
148     59   1:normal   1:normal         *0.999  0.001 (5)
149     60   1:normal   1:normal         *0.999  0.001 (5)
150     61   1:normal   1:normal         *0.999  0.001 (5)
151     62   1:normal   1:normal         *0.999  0.001 (5)
152     63   1:normal   1:normal         *0.999  0.001 (5)
153     64   1:normal   1:normal         *0.999  0.001 (5)
154     65   1:normal   1:normal         *0.999  0.001 (5)
155     66   1:normal   1:normal         *0.999  0.001 (5)
156     67   1:normal   1:normal         *0.999  0.001 (5)
157     68   1:normal   1:normal         *0.999  0.001 (5)
158     69   1:normal   1:normal         *0.986  0.014 (5)
159     70   1:normal   1:normal         *0.999  0.001 (5)
160     71   1:normal   1:normal         *0.999  0.001 (5)
161     72   1:normal   1:normal         *0.999  0.001 (5)
162     73   1:normal   1:normal         *0.999  0.001 (5)
163     74   1:normal   1:normal         *0.999  0.001 (5)
164     75   1:normal   1:normal         *1      0     (5)
165     76   1:normal   1:normal         *0.986  0.014 (5)
166     77   1:normal   1:normal         *0.999  0.001 (5)
167     78   1:normal   1:normal         *0.999  0.001 (5)
168     79   1:normal   1:normal         *0.999  0.001 (5)
169     80   1:normal   1:normal         *0.999  0.001 (5)
170     81   1:normal   1:normal         *0.986  0.014 (5)
171     82   1:normal   1:normal         *0.999  0.001 (5)
172     83   1:normal   1:normal         *0.986  0.014 (5)
173     84   1:normal   1:normal         *0.999  0.001 (5)
174     85   1:normal   1:normal         *0.986  0.014 (5)
175     86   1:normal   1:normal         *0.999  0.001 (5)
176     87   1:normal   1:normal         *1      0     (5)
177     88   1:normal   1:normal         *0.999  0.001 (5)
178     89   1:normal   1:normal         *0.999  0.001 (5)
179     90   1:normal   1:normal         *0.999  0.001 (5)
180     91   1:normal   1:normal         *0.999  0.001 (5)
181     92   1:normal   1:normal         *0.999  0.001 (5)
182     93   1:normal   1:normal         *0.999  0.001 (5)
```

```
183    94   1:normal   1:normal        *0.999  0.001 (5)
184    95   1:normal   1:normal        *0.999  0.001 (5)
185    96   1:normal   1:normal        *0.999  0.001 (5)
186    97   1:normal   1:normal        *0.986  0.014 (5)
187    98   1:normal   1:normal        *0.999  0.001 (5)
188    99   1:normal   1:normal        *0.999  0.001 (5)
189   100   1:normal   1:normal        *0.999  0.001 (5)
190   101   1:normal   1:normal        *0.999  0.001 (5)
191   102   1:normal   1:normal        *0.999  0.001 (5)
192   103   1:normal   1:normal        *0.986  0.014 (5)
193   104   1:normal   1:normal        *0.999  0.001 (5)
194   105   1:normal   1:normal        *0.999  0.001 (5)
195   106   1:normal   1:normal        *0.999  0.001 (5)
196   107   1:normal   1:normal        *0.999  0.001 (5)
197   108   1:normal   1:normal        *0.999  0.001 (5)
198   109   1:normal   1:normal        *0.999  0.001 (5)
199   110   1:normal   1:normal        *0.999  0.001 (5)
200   111   1:normal   1:normal        *0.999  0.001 (5)
201   112   1:normal   1:normal        *0.999  0.001 (5)
202   113   1:normal   1:normal        *0.999  0.001 (5)
203   114   1:normal   1:normal        *0.999  0.001 (5)
204   115   1:normal   1:normal        *0.999  0.001 (5)
205   116   1:normal   1:normal        *0.999  0.001 (5)
206   117   1:normal   1:normal        *0.999  0.001 (5)
207   118   1:normal   1:normal        *0.999  0.001 (5)
208   119   1:normal   1:normal        *0.999  0.001 (5)
209   120   1:normal   1:normal        *0.999  0.001 (5)
210   121   1:normal   1:normal        *0.999  0.001 (5)
211   122   1:normal   1:normal        *0.999  0.001 (5)
212   123   1:normal   1:normal        *0.999  0.001 (5)
213   124   1:normal   1:normal        *0.999  0.001 (5)
214   125   1:normal   1:normal        *0.999  0.001 (5)
215   126   1:normal   1:normal        *0.999  0.001 (5)
216   127   1:normal   1:normal        *0.999  0.001 (5)
217   128   1:normal   1:normal        *0.999  0.001 (5)
```

```
218   129   1:normal   1:normal         *0.999  0.001 (5)
219   130   1:normal   1:normal         *0.999  0.001 (5)
220   131   1:normal   1:normal         *0.999  0.001 (5)
221   132   1:normal   1:normal         *0.999  0.001 (5)
222   133   1:normal   1:normal         *0.999  0.001 (5)
223   134   1:normal   1:normal         *0.999  0.001 (5)
224   135   1:normal   1:normal         *0.999  0.001 (5)
225   136   1:normal   1:normal         *0.999  0.001 (5)
226   137   1:normal   1:normal         *0.999  0.001 (5)
227   138   1:normal   1:normal         *0.999  0.001 (5)
228   139   1:normal   1:normal         *0.999  0.001 (5)
229   140   1:normal   1:normal         *0.999  0.001 (5)
230   141   1:normal   1:normal         *0.999  0.001 (5)
231   142   1:normal   1:normal         *0.999  0.001 (5)
232   143   1:normal   1:normal         *0.999  0.001 (5)
233   144   1:normal   1:normal         *0.999  0.001 (5)
234   145   1:normal   1:normal         *0.999  0.001 (5)
235   146   1:normal   1:normal         *0.999  0.001 (5)
236   147   1:normal   1:normal         *0.986  0.014 (5)
237   148   1:normal   1:normal         *0.999  0.001 (5)
238   149   1:normal   1:normal         *1      0     (5)
239   150   1:normal   1:normal         *0.999  0.001 (5)
240   151   1:normal   1:normal         *1      0     (5)
241   152   1:normal   1:normal         *0.999  0.001 (5)
242   153   1:normal   1:normal         *0.999  0.001 (5)
243   154   1:normal   1:normal         *0.999  0.001 (5)
244   155   1:normal   1:normal         *0.986  0.014 (5)
245   156   1:normal   1:normal         *0.999  0.001 (5)
246   157   1:normal   1:normal         *0.999  0.001 (5)
247   158   1:normal   1:normal         *1      0     (5)
248   159   1:normal   1:normal         *1      0     (5)
249   160   1:normal   1:normal         *0.999  0.001 (5)
250   161   1:normal   1:normal         *0.999  0.001 (5)
251   162   1:normal   1:normal         *0.999  0.001 (5)
252   163   1:normal   1:normal         *0.999  0.001 (5)
```

```
253   164   1:normal   1:normal          *0.986  0.014 (5)
254   165   1:normal   1:normal          *0.999  0.001 (5)
255   166   1:normal   1:normal          *0.999  0.001 (5)
256   167   1:normal   1:normal          *0.999  0.001 (5)
257   168   1:normal   1:normal          *0.999  0.001 (5)
258   169   1:normal   1:normal          *0.999  0.001 (5)
259   170   1:normal   1:normal          *0.999  0.001 (5)
260   171   1:normal   1:normal          *0.999  0.001 (5)
261   172   1:normal   1:normal          *0.999  0.001 (5)
262   173   1:normal   1:normal          *0.999  0.001 (5)
263   174   1:normal   1:normal          *0.999  0.001 (5)
264   175   1:normal   1:normal          *0.999  0.001 (5)
265   176   1:normal   1:normal          *0.999  0.001 (5)
266   177  2:anomalo   1:normal      +   *0.986  0.014 (5)
267   178   1:normal   1:normal          *0.999  0.001 (5)
268   179   1:normal   1:normal          *0.999  0.001 (5)
269   180  2:anomalo   1:normal      +   *0.986  0.014 (5)
270   181   1:normal   1:normal          *0.999  0.001 (5)
271   182   1:normal   1:normal          *0.999  0.001 (5)
272   183   1:normal   1:normal          *0.999  0.001 (5)
273   184   1:normal   1:normal          *0.999  0.001 (5)
274   185   1:normal   1:normal          *0.999  0.001 (5)
275   186   1:normal   1:normal          *0.999  0.001 (5)
276   187   1:normal   1:normal          *0.999  0.001 (5)
277   188   1:normal   1:normal          *0.999  0.001 (5)
278   189   1:normal   1:normal          *0.999  0.001 (5)
279   190   1:normal   1:normal          *0.999  0.001 (5)
280   191   1:normal   1:normal          *0.999  0.001 (5)
281   192   1:normal   1:normal          *0.999  0.001 (5)
282   193   1:normal   1:normal          *0.999  0.001 (5)
283   194   1:normal   1:normal          *0.999  0.001 (5)
284   195   1:normal   1:normal          *0.999  0.001 (5)
285   196   1:normal   1:normal          *0.999  0.001 (5)
286   197   1:normal   1:normal          *0.999  0.001 (5)
287   198   1:normal   1:normal          *0.999  0.001 (5)
```

```
288    199   1:normal   1:normal          *0.999  0.001 (5)
289    200   1:normal   1:normal          *0.999  0.001 (5)
290    201   1:normal   1:normal          *0.999  0.001 (5)
291    202   1:normal   1:normal          *0.999  0.001 (5)
292    203   1:normal   1:normal          *0.999  0.001 (5)
293    204   1:normal   1:normal          *0.999  0.001 (5)
294    205   1:normal   1:normal          *0.999  0.001 (5)
295    206   1:normal   1:normal          *1      0     (5)
296    207   1:normal   1:normal          *0.999  0.001 (5)
297    208   1:normal   1:normal          *0.999  0.001 (5)
298    209   1:normal   1:normal          *0.999  0.001 (5)
299    210   1:normal   1:normal          *0.999  0.001 (5)
300    211   1:normal   1:normal          *0.999  0.001 (5)
301    212   1:normal   1:normal          *0.999  0.001 (5)
302    213  2:anomalo  2:anomalo           0     *1     (5)
303    214   1:normal   1:normal          *0.999  0.001 (5)
304    215  2:anomalo   1:normal      +   *0.999  0.001 (5)
305    216   1:normal   1:normal          *0.999  0.001 (5)
306    217   1:normal   1:normal          *0.999  0.001 (5)
307    218   1:normal   1:normal          *0.999  0.001 (5)
308    219   1:normal   1:normal          *0.999  0.001 (5)
309    220   1:normal   1:normal          *0.999  0.001 (5)
310    221   1:normal   1:normal          *0.999  0.001 (5)
311    222   1:normal   1:normal          *0.999  0.001 (5)
312    223   1:normal   1:normal          *0.999  0.001 (5)
313    224   1:normal   1:normal          *0.999  0.001 (5)
314    225   1:normal   1:normal          *0.999  0.001 (5)
315    226   1:normal   1:normal          *0.999  0.001 (5)
316    227   1:normal   1:normal          *0.999  0.001 (5)
317    228   1:normal   1:normal          *0.999  0.001 (5)
318    229   1:normal   1:normal          *0.999  0.001 (5)
319    230   1:normal   1:normal          *0.999  0.001 (5)
320    231   1:normal   1:normal          *0.999  0.001 (5)
321    232   1:normal   1:normal          *0.999  0.001 (5)
322    233   1:normal   1:normal          *0.999  0.001 (5)
```

```
323   234   1:normal   1:normal        *0.999  0.001 (5)
324   235   1:normal   1:normal        *0.999  0.001 (5)
325   236   1:normal   1:normal        *0.999  0.001 (5)
326   237   1:normal   1:normal        *0.999  0.001 (5)
327   238   1:normal   1:normal        *0.999  0.001 (5)
328   239   1:normal   1:normal        *0.999  0.001 (5)
329   240   1:normal   1:normal        *0.999  0.001 (5)
330   241   1:normal   1:normal        *0.999  0.001 (5)
331   242   1:normal   1:normal        *0.999  0.001 (5)
332   243   1:normal   1:normal        *0.999  0.001 (5)
333   244   1:normal   1:normal        *0.999  0.001 (5)
334   245   1:normal   1:normal        *0.999  0.001 (5)
335   246   1:normal   1:normal        *0.999  0.001 (5)
336   247   1:normal   1:normal        *0.999  0.001 (5)
337   248   1:normal   1:normal        *0.999  0.001 (5)
338   249   1:normal   1:normal        *0.999  0.001 (5)
339   250   1:normal   1:normal        *0.999  0.001 (5)
340   251   1:normal   1:normal        *0.999  0.001 (5)
341   252   1:normal   1:normal        *0.999  0.001 (5)
342   253   1:normal   1:normal        *0.999  0.001 (5)
343   254   1:normal   1:normal        *0.999  0.001 (5)
344   255   1:normal   1:normal        *0.999  0.001 (5)
345   256   1:normal   1:normal        *0.999  0.001 (5)
346   257   1:normal   1:normal        *0.999  0.001 (5)
347   258   1:normal   1:normal        *0.999  0.001 (5)
348   259   1:normal   1:normal        *0.999  0.001 (5)
349   260   1:normal   1:normal        *0.999  0.001 (5)
350   261   1:normal   1:normal        *0.999  0.001 (5)
351   262   1:normal   1:normal        *0.999  0.001 (5)
352   263   1:normal   1:normal        *0.999  0.001 (5)
353   264   1:normal   1:normal        *0.999  0.001 (5)
354   265   1:normal   1:normal        *0.999  0.001 (5)
355   266   1:normal   1:normal        *0.999  0.001 (5)
356   267   1:normal   1:normal        *0.999  0.001 (5)
357   268   1:normal   1:normal        *0.999  0.001 (5)
```

```
358    269   1:normal   1:normal          *0.999  0.001 (5)
359    270   1:normal   1:normal          *0.999  0.001 (5)
360    271   1:normal   1:normal          *0.999  0.001 (5)
361    272   1:normal   1:normal          *0.999  0.001 (5)
362    273   1:normal   1:normal          *0.999  0.001 (5)
363    274   1:normal   1:normal          *0.999  0.001 (5)
364    275   1:normal   1:normal          *0.999  0.001 (5)
365    276   1:normal   1:normal          *0.999  0.001 (5)
366    277   1:normal   1:normal          *0.999  0.001 (5)
367    278   1:normal   1:normal          *0.999  0.001 (5)
368    279   1:normal   1:normal          *0.999  0.001 (5)
369    280   1:normal   1:normal          *0.999  0.001 (5)
370    281   1:normal   1:normal          *0.999  0.001 (5)
371    282   1:normal   1:normal          *0.999  0.001 (5)
372    283   1:normal   1:normal          *0.999  0.001 (5)
373    284   1:normal   1:normal          *0.999  0.001 (5)
374    285   1:normal   1:normal          *0.999  0.001 (5)
375    286   1:normal   1:normal          *0.999  0.001 (5)
376    287   1:normal   1:normal          *0.999  0.001 (5)
377    288   1:normal   1:normal          *1      0     (5)
378    289   1:normal   1:normal          *1      0     (5)
379    290   1:normal   1:normal          *0.999  0.001 (5)
380    291   1:normal   1:normal          *0.999  0.001 (5)
381    292   1:normal   1:normal          *0.999  0.001 (5)
382    293   1:normal   1:normal          *0.999  0.001 (5)
383    294   1:normal   1:normal          *0.999  0.001 (5)
384    295   1:normal   1:normal          *0.999  0.001 (5)
385    296   1:normal   1:normal          *0.999  0.001 (5)
386    297   1:normal   1:normal          *0.999  0.001 (5)
387    298   1:normal   1:normal          *0.999  0.001 (5)
388    299  2:anomalo  2:anomalo           0     *1     (5)
389    300   1:normal   1:normal          *0.999  0.001 (5)
390    301  2:anomalo   1:normal      +   *0.999  0.001 (5)
391    302   1:normal   1:normal          *0.999  0.001 (5)
392    303   1:normal   1:normal          *0.999  0.001 (5)
```

```
393   304   1:normal   1:normal        *0.999  0.001 (5)
394   305   1:normal   1:normal        *0.999  0.001 (5)
395   306   1:normal   1:normal        *0.999  0.001 (5)
396   307   1:normal   1:normal        *0.999  0.001 (5)
397   308   1:normal   1:normal        *0.999  0.001 (5)
398   309   1:normal   1:normal        *0.999  0.001 (5)
399   310   1:normal   1:normal        *0.999  0.001 (5)
400   311   1:normal   1:normal        *0.999  0.001 (5)
401   312   1:normal   1:normal        *0.999  0.001 (5)
402   313   1:normal   1:normal        *0.999  0.001 (5)
403   314   1:normal   1:normal        *0.999  0.001 (5)
404   315   1:normal   1:normal        *0.999  0.001 (5)
405   316   1:normal   1:normal        *0.999  0.001 (5)
406   317   1:normal   1:normal        *0.999  0.001 (5)
407   318   1:normal   1:normal        *0.999  0.001 (5)
408   319   1:normal   1:normal        *0.999  0.001 (5)
409   320   1:normal   1:normal        *0.999  0.001 (5)
410   321   1:normal   1:normal        *0.999  0.001 (5)
411   322   1:normal   1:normal        *0.999  0.001 (5)
412   323   1:normal   1:normal        *0.999  0.001 (5)
413   324   1:normal   1:normal        *0.999  0.001 (5)
414   325   1:normal   1:normal        *0.999  0.001 (5)
415   326   1:normal   1:normal        *0.999  0.001 (5)
416   327   1:normal   1:normal        *0.999  0.001 (5)
417   328   1:normal   1:normal        *0.999  0.001 (5)
418   329   1:normal   1:normal        *0.999  0.001 (5)
419   330   1:normal   1:normal        *0.999  0.001 (5)
420   331   1:normal   1:normal        *0.999  0.001 (5)
421   332   1:normal   1:normal        *0.999  0.001 (5)
422   333   1:normal   1:normal        *0.999  0.001 (5)
423   334   1:normal   1:normal        *0.999  0.001 (5)
424   335   1:normal   1:normal        *0.999  0.001 (5)
425   336   1:normal   1:normal        *0.999  0.001 (5)
426   337   1:normal   1:normal        *0.999  0.001 (5)
427   338   1:normal   1:normal        *0.999  0.001 (5)
```

```
428   339   1:normal   1:normal         *0.999  0.001 (5)
429   340   1:normal   1:normal         *0.999  0.001 (5)
430   341   1:normal   1:normal         *0.999  0.001 (5)
431   342   1:normal   1:normal         *0.999  0.001 (5)
432   343   1:normal   1:normal         *0.999  0.001 (5)
433   344   1:normal   1:normal         *0.999  0.001 (5)
434   345   1:normal   1:normal         *0.999  0.001 (5)
435   346   1:normal   1:normal         *0.999  0.001 (5)
436   347   1:normal   1:normal         *0.999  0.001 (5)
437   348   1:normal   1:normal         *0.999  0.001 (5)
438   349   1:normal   1:normal         *0.999  0.001 (5)
439   350   1:normal   1:normal         *0.999  0.001 (5)
440   351   1:normal   1:normal         *0.999  0.001 (5)
441   352   1:normal   1:normal         *0.999  0.001 (5)
442   353   1:normal   1:normal         *0.999  0.001 (5)
443   354   1:normal   1:normal         *0.999  0.001 (5)
444   355   1:normal   1:normal         *0.999  0.001 (5)
445   356   1:normal   1:normal         *0.999  0.001 (5)
446   357   1:normal   1:normal         *0.999  0.001 (5)
447   358   1:normal   1:normal         *0.999  0.001 (5)
448   359   1:normal   1:normal         *0.999  0.001 (5)
449   360   1:normal   1:normal         *0.999  0.001 (5)
450   361   1:normal   1:normal         *0.999  0.001 (5)
451   362   1:normal   1:normal         *0.999  0.001 (5)
452   363   1:normal   1:normal         *0.999  0.001 (5)
453   364   1:normal   1:normal         *0.999  0.001 (5)
454   365   1:normal   1:normal         *0.999  0.001 (5)
455   366   1:normal   1:normal         *0.999  0.001 (5)
456   367   1:normal   1:normal         *0.999  0.001 (5)
457   368   1:normal   1:normal         *0.999  0.001 (5)
458   369   1:normal   1:normal         *0.999  0.001 (5)
459   370   1:normal   1:normal         *0.999  0.001 (5)
460   371   1:normal   1:normal         *0.999  0.001 (5)
461   372   1:normal   1:normal         *0.999  0.001 (5)
462   373   1:normal   1:normal         *0.999  0.001 (5)
```

```
463   374   1:normal   1:normal        *0.999  0.001 (5)
464   375   1:normal   1:normal        *0.999  0.001 (5)
465   376   1:normal   1:normal        *0.999  0.001 (5)
466   377   1:normal   1:normal        *0.999  0.001 (5)
467   378   1:normal   1:normal        *0.999  0.001 (5)
468   379   1:normal   1:normal        *0.999  0.001 (5)
469   380   1:normal   1:normal        *0.999  0.001 (5)
470   381   1:normal   1:normal        *0.999  0.001 (5)
471   382   1:normal   1:normal        *0.999  0.001 (5)
472   383   1:normal   1:normal        *0.999  0.001 (5)
473   384   1:normal   1:normal        *0.999  0.001 (5)
474   385   1:normal   1:normal        *0.999  0.001 (5)
475   386   1:normal   1:normal        *0.999  0.001 (5)
476   387   1:normal   1:normal        *0.999  0.001 (5)
477   388   1:normal   1:normal        *0.999  0.001 (5)
478   389   1:normal   1:normal        *0.999  0.001 (5)
479   390   1:normal   1:normal        *0.999  0.001 (5)
480   391   1:normal   1:normal        *0.999  0.001 (5)
481   392   1:normal   1:normal        *0.999  0.001 (5)
482   393   1:normal   1:normal        *0.999  0.001 (5)
483   394   1:normal   1:normal        *0.999  0.001 (5)
484   395   1:normal   1:normal        *0.999  0.001 (5)
485   396   1:normal   1:normal        *0.999  0.001 (5)
486   397   1:normal   1:normal        *0.999  0.001 (5)
487   398   1:normal   1:normal        *0.999  0.001 (5)
488   399   1:normal   1:normal        *0.999  0.001 (5)
489   400   1:normal   1:normal        *0.999  0.001 (5)
490   401   1:normal   1:normal        *0.999  0.001 (5)
491   402   1:normal   1:normal        *0.999  0.001 (5)
492   403   1:normal   1:normal        *0.999  0.001 (5)
493   404   1:normal   1:normal        *0.999  0.001 (5)
494   405   1:normal   1:normal        *0.999  0.001 (5)
495   406   1:normal   1:normal        *0.999  0.001 (5)
496   407   1:normal   1:normal        *0.999  0.001 (5)
497   408   1:normal   1:normal        *0.999  0.001 (5)
```

```
498    409   1:normal   1:normal          *0.999  0.001 (5)
499    410   1:normal   1:normal          *0.999  0.001 (5)
500    411   1:normal   1:normal          *0.999  0.001 (5)
501    412   1:normal   1:normal          *0.999  0.001 (5)
502    413  2:anomalo  2:anomalo           0      *1      (5)
503    414   1:normal   1:normal          *0.999  0.001 (5)
504    415   1:normal   1:normal          *0.999  0.001 (5)
505    416   1:normal   1:normal          *0.999  0.001 (5)
506    417   1:normal   1:normal          *0.999  0.001 (5)
507    418   1:normal   1:normal          *0.999  0.001 (5)
508    419   1:normal   1:normal          *0.999  0.001 (5)
509    420   1:normal   1:normal          *0.999  0.001 (5)
510    421   1:normal   1:normal          *0.999  0.001 (5)
511    422   1:normal   1:normal          *0.999  0.001 (5)
512    423   1:normal   1:normal          *0.999  0.001 (5)
513    424  2:anomalo  2:anomalo           0      *1      (5)
514    425   1:normal   1:normal          *0.999  0.001 (5)
515    426   1:normal   1:normal          *0.999  0.001 (5)
516    427  2:anomalo   1:normal      +    *0.999  0.001 (5)
517    428   1:normal   1:normal          *0.999  0.001 (5)
518    429   1:normal   1:normal          *0.999  0.001 (5)
519    430   1:normal   1:normal          *0.999  0.001 (5)
520    431   1:normal   1:normal          *0.999  0.001 (5)
521    432   1:normal   1:normal          *0.999  0.001 (5)
522    433   1:normal   1:normal          *0.999  0.001 (5)
523    434   1:normal   1:normal          *0.999  0.001 (5)
524    435   1:normal   1:normal          *0.999  0.001 (5)
525    436   1:normal   1:normal          *0.999  0.001 (5)
526    437   1:normal   1:normal          *0.999  0.001 (5)
527    438   1:normal   1:normal          *0.999  0.001 (5)
528    439   1:normal   1:normal          *0.999  0.001 (5)
529    440   1:normal   1:normal          *0.999  0.001 (5)
530    441   1:normal   1:normal          *0.999  0.001 (5)
531    442   1:normal   1:normal          *0.999  0.001 (5)
532    443   1:normal   1:normal          *0.999  0.001 (5)
```

```
533   444   1:normal   1:normal        *0.999  0.001 (5)
534   445   1:normal   1:normal        *0.999  0.001 (5)
535   446   1:normal   1:normal        *0.999  0.001 (5)
536   447   1:normal   1:normal        *0.999  0.001 (5)
537   448   1:normal   1:normal        *0.999  0.001 (5)
538   449   1:normal   1:normal        *0.999  0.001 (5)
539   450   1:normal   1:normal        *0.999  0.001 (5)
540   451   1:normal   1:normal        *0.999  0.001 (5)
541   452   1:normal   1:normal        *0.999  0.001 (5)
542   453   1:normal   1:normal        *0.999  0.001 (5)
543   454   1:normal   1:normal        *0.999  0.001 (5)
544   455   1:normal   1:normal        *0.999  0.001 (5)
545   456   1:normal   1:normal        *0.999  0.001 (5)
546   457   1:normal   1:normal        *0.999  0.001 (5)
547   458   1:normal   1:normal        *0.999  0.001 (5)
548   459   1:normal   1:normal        *0.999  0.001 (5)
549   460   1:normal   1:normal        *0.999  0.001 (5)
550   461   1:normal   1:normal        *0.999  0.001 (5)
551   462   1:normal   1:normal        *0.999  0.001 (5)
552   463   1:normal   1:normal        *0.999  0.001 (5)
553   464   1:normal   1:normal        *0.999  0.001 (5)
554   465   1:normal   1:normal        *0.999  0.001 (5)
555   466   1:normal   1:normal        *0.999  0.001 (5)
556   467   1:normal   1:normal        *0.999  0.001 (5)
557   468   1:normal   1:normal        *0.999  0.001 (5)
558   469   1:normal   1:normal        *0.999  0.001 (5)
559   470   1:normal   1:normal        *0.999  0.001 (5)
560   471   1:normal   1:normal        *0.999  0.001 (5)
561   472   1:normal   1:normal        *0.999  0.001 (5)
562   473   1:normal   1:normal        *0.999  0.001 (5)
563   474   1:normal   1:normal        *0.999  0.001 (5)
564   475   1:normal   1:normal        *0.999  0.001 (5)
565   476   1:normal   1:normal        *0.999  0.001 (5)
566   477   1:normal   1:normal        *0.999  0.001 (5)
567   478   1:normal   1:normal        *0.999  0.001 (5)
```

```
568   479   1:normal   1:normal         *0.999  0.001 (5)
569   480   1:normal   1:normal         *0.999  0.001 (5)
570   481   1:normal   1:normal         *0.999  0.001 (5)
571   482   1:normal   1:normal         *0.999  0.001 (5)
572   483   1:normal   1:normal         *0.999  0.001 (5)
573   484   1:normal   1:normal         *0.999  0.001 (5)
574   485   1:normal   1:normal         *0.999  0.001 (5)
575   486   1:normal   1:normal         *0.999  0.001 (5)
576   487   1:normal   1:normal         *0.999  0.001 (5)
577   488   1:normal   1:normal         *0.999  0.001 (5)
578   489   1:normal   1:normal         *0.999  0.001 (5)
579   490   1:normal   1:normal         *0.999  0.001 (5)
580   491   1:normal   1:normal         *0.999  0.001 (5)
581   492   1:normal   1:normal         *0.999  0.001 (5)
582   493   1:normal   1:normal         *0.999  0.001 (5)
583   494   1:normal   1:normal         *0.999  0.001 (5)
584   495   1:normal   1:normal         *0.999  0.001 (5)
585   496   1:normal   1:normal         *0.999  0.001 (5)
586   497   1:normal   1:normal         *0.999  0.001 (5)
587   498   1:normal   1:normal         *0.999  0.001 (5)
588   499   1:normal   1:normal         *0.999  0.001 (5)
589   500   1:normal   1:normal         *0.999  0.001 (5)
590   501   1:normal   1:normal         *0.999  0.001 (5)
591   502   1:normal   1:normal         *0.999  0.001 (5)
592   503   1:normal   1:normal         *0.999  0.001 (5)
593   504   1:normal   1:normal         *0.999  0.001 (5)
594   505   1:normal   1:normal         *0.999  0.001 (5)
595   506   1:normal   1:normal         *0.999  0.001 (5)
596   507   1:normal   1:normal         *0.999  0.001 (5)
597   508   1:normal   1:normal         *0.999  0.001 (5)
598   509   1:normal   1:normal         *0.999  0.001 (5)
599   510   1:normal   1:normal         *0.999  0.001 (5)
600   511   1:normal   1:normal         *0.999  0.001 (5)
601   512   1:normal   1:normal         *0.999  0.001 (5)
602   513   1:normal   1:normal         *0.999  0.001 (5)
```

```
603    514   1:normal   1:normal          *0.999  0.001 (5)
604    515   1:normal   1:normal          *0.999  0.001 (5)
605    516   1:normal   1:normal          *0.999  0.001 (5)
606    517   1:normal   1:normal          *0.999  0.001 (5)
607    518   1:normal   1:normal          *0.999  0.001 (5)
608    519   1:normal   1:normal          *0.999  0.001 (5)
609    520   1:normal   1:normal          *0.999  0.001 (5)
610    521   1:normal   1:normal          *0.999  0.001 (5)
611    522   1:normal   1:normal          *0.999  0.001 (5)
612    523   1:normal   1:normal          *0.999  0.001 (5)
613    524   1:normal   1:normal          *0.999  0.001 (5)
614    525   1:normal   1:normal          *0.999  0.001 (5)
615    526   1:normal   1:normal          *0.999  0.001 (5)
616    527   1:normal   1:normal          *0.999  0.001 (5)
617    528   1:normal   1:normal          *0.999  0.001 (5)
618    529   1:normal   1:normal          *0.999  0.001 (5)
619    530   1:normal   1:normal          *0.999  0.001 (5)
620    531   1:normal   1:normal          *0.999  0.001 (5)
621    532   1:normal   1:normal          *0.999  0.001 (5)
622    533   1:normal   1:normal          *0.999  0.001 (5)
623    534  2:anomalo  2:anomalo           0     *1     (5)
624    535   1:normal   1:normal          *0.999  0.001 (5)
625    536  2:anomalo   1:normal      +   *0.999  0.001 (5)
626    537   1:normal   1:normal          *0.999  0.001 (5)
627    538   1:normal   1:normal          *0.999  0.001 (5)
628    539   1:normal   1:normal          *0.999  0.001 (5)
629    540   1:normal   1:normal          *0.999  0.001 (5)
630    541   1:normal   1:normal          *0.999  0.001 (5)
631    542   1:normal   1:normal          *0.999  0.001 (5)
632    543   1:normal   1:normal          *0.999  0.001 (5)
633    544   1:normal   1:normal          *0.999  0.001 (5)
634    545   1:normal   1:normal          *0.999  0.001 (5)
635    546   1:normal   1:normal          *0.999  0.001 (5)
636    547   1:normal   1:normal          *0.999  0.001 (5)
637    548   1:normal   1:normal          *0.999  0.001 (5)
```

```
638    549   1:normal   1:normal         *0.999  0.001 (5)
639    550   1:normal   1:normal         *0.999  0.001 (5)
640    551   1:normal   1:normal         *0.999  0.001 (5)
641    552   1:normal   1:normal         *0.999  0.001 (5)
642    553   1:normal   1:normal         *0.999  0.001 (5)
643    554   1:normal   1:normal         *0.999  0.001 (5)
644    555   1:normal   1:normal         *0.999  0.001 (5)
645    556   1:normal   1:normal         *0.999  0.001 (5)
646    557   1:normal   1:normal         *0.999  0.001 (5)
647    558   1:normal   1:normal         *0.999  0.001 (5)
648    559   1:normal   1:normal         *0.999  0.001 (5)
649    560   1:normal   1:normal         *0.999  0.001 (5)
650    561   1:normal   1:normal         *0.999  0.001 (5)
651    562   1:normal   1:normal         *0.999  0.001 (5)
652    563   1:normal   1:normal         *0.999  0.001 (5)
653    564   1:normal   1:normal         *0.999  0.001 (5)
654    565   1:normal   1:normal         *0.999  0.001 (5)
655    566   1:normal   1:normal         *0.999  0.001 (5)
656    567   1:normal   1:normal         *0.999  0.001 (5)
657    568   1:normal   1:normal         *0.999  0.001 (5)
658    569   1:normal   1:normal         *0.999  0.001 (5)
659    570   1:normal   1:normal         *0.999  0.001 (5)
660    571   1:normal   1:normal         *0.999  0.001 (5)
661    572   1:normal   1:normal         *0.999  0.001 (5)
662    573   1:normal   1:normal         *0.999  0.001 (5)
663    574   1:normal   1:normal         *0.999  0.001 (5)
664    575   1:normal   1:normal         *0.999  0.001 (5)
665    576   1:normal   1:normal         *0.999  0.001 (5)
666    577   1:normal   1:normal         *0.999  0.001 (5)
667    578   1:normal   1:normal         *0.999  0.001 (5)
668    579   1:normal   1:normal         *0.999  0.001 (5)
669    580   1:normal   1:normal         *0.999  0.001 (5)
670    581   1:normal   1:normal         *0.999  0.001 (5)
671    582   1:normal   1:normal         *0.999  0.001 (5)
672    583   1:normal   1:normal         *0.999  0.001 (5)
```

```
673   584   1:normal   1:normal        *0.999  0.001 (5)
674   585   1:normal   1:normal        *0.999  0.001 (5)
675   586   1:normal   1:normal        *0.999  0.001 (5)
676   587   1:normal   1:normal        *0.999  0.001 (5)
677   588   1:normal   1:normal        *0.999  0.001 (5)
678   589   1:normal   1:normal        *0.999  0.001 (5)
679   590   1:normal   1:normal        *0.999  0.001 (5)
680   591   1:normal   1:normal        *0.999  0.001 (5)
681   592   1:normal   1:normal        *0.999  0.001 (5)
682   593   1:normal   1:normal        *0.999  0.001 (5)
683   594   1:normal   1:normal        *0.999  0.001 (5)
684   595   1:normal   1:normal        *0.999  0.001 (5)
685   596   1:normal   1:normal        *0.999  0.001 (5)
686   597   1:normal   1:normal        *0.999  0.001 (5)
687   598   1:normal   1:normal        *0.999  0.001 (5)
688   599   1:normal   1:normal        *0.999  0.001 (5)
689   600   1:normal   1:normal        *0.999  0.001 (5)
690   601   1:normal   1:normal        *0.999  0.001 (5)
691   602   1:normal   1:normal        *0.999  0.001 (5)
692   603   1:normal   1:normal        *0.999  0.001 (5)
693   604   1:normal   1:normal        *0.999  0.001 (5)
694   605   1:normal   1:normal        *0.999  0.001 (5)
695   606   1:normal   1:normal        *0.999  0.001 (5)
696   607   1:normal   1:normal        *0.999  0.001 (5)
697   608   1:normal   1:normal        *0.999  0.001 (5)
698   609   1:normal   1:normal        *0.999  0.001 (5)
699   610   1:normal   1:normal        *0.999  0.001 (5)
700   611   1:normal   1:normal        *0.999  0.001 (5)
701   612   1:normal   1:normal        *0.999  0.001 (5)
702   613   1:normal   1:normal        *0.999  0.001 (5)
703   614   1:normal   1:normal        *0.999  0.001 (5)
704   615   1:normal   1:normal        *0.999  0.001 (5)
705   616   1:normal   1:normal        *0.999  0.001 (5)
706   617   1:normal   1:normal        *0.999  0.001 (5)
707   618   1:normal   1:normal        *0.999  0.001 (5)
```

```
708    619   1:normal   1:normal         *0.999  0.001 (5)
709    620   1:normal   1:normal         *0.999  0.001 (5)
710    621   1:normal   1:normal         *0.999  0.001 (5)
711    622   1:normal   1:normal         *0.999  0.001 (5)
712    623   1:normal   1:normal         *0.999  0.001 (5)
713    624   1:normal   1:normal         *0.999  0.001 (5)
714    625   1:normal   1:normal         *0.999  0.001 (5)
715    626   1:normal   1:normal         *0.999  0.001 (5)
716    627   1:normal   1:normal         *0.999  0.001 (5)
717    628   1:normal   1:normal         *0.999  0.001 (5)
718    629   1:normal   1:normal         *0.999  0.001 (5)
719    630   1:normal   1:normal         *0.999  0.001 (5)
720    631   1:normal   1:normal         *0.999  0.001 (5)
721    632   1:normal   1:normal         *0.999  0.001 (5)
722    633   1:normal   1:normal         *0.999  0.001 (5)
723    634   1:normal   1:normal         *0.999  0.001 (5)
724    635   1:normal   1:normal         *0.999  0.001 (5)
725    636   1:normal   1:normal         *0.999  0.001 (5)
726    637   1:normal   1:normal         *0.999  0.001 (5)
727    638   1:normal   1:normal         *0.999  0.001 (5)
728    639   1:normal   1:normal         *0.999  0.001 (5)
729    640   1:normal   1:normal         *0.999  0.001 (5)
730    641   1:normal   1:normal         *0.999  0.001 (5)
731    642   1:normal   1:normal         *0.999  0.001 (5)
732    643   1:normal   1:normal         *0.999  0.001 (5)
733    644   1:normal   1:normal         *0.999  0.001 (5)
734    645   1:normal   1:normal         *0.999  0.001 (5)
735    646   1:normal   1:normal         *0.999  0.001 (5)
736    647   1:normal   1:normal         *0.999  0.001 (5)
737    648   1:normal   1:normal         *0.999  0.001 (5)
738    649   1:normal   1:normal         *0.999  0.001 (5)
739    650   1:normal   1:normal         *0.999  0.001 (5)
740    651   1:normal   1:normal         *0.999  0.001 (5)
741    652   1:normal   1:normal         *0.999  0.001 (5)
742    653   1:normal   1:normal         *0.999  0.001 (5)
```

```
743   654   1:normal   1:normal        *0.999  0.001 (5)
744   655   1:normal   1:normal        *0.999  0.001 (5)
745   656   1:normal   1:normal        *0.999  0.001 (5)
746   657   1:normal   1:normal        *0.999  0.001 (5)
747   658   1:normal   1:normal        *0.999  0.001 (5)
748   659   1:normal   1:normal        *0.999  0.001 (5)
749   660   1:normal   1:normal        *0.999  0.001 (5)
750   661   1:normal   1:normal        *0.999  0.001 (5)
751   662   1:normal   1:normal        *0.999  0.001 (5)
752   663   1:normal   1:normal        *0.999  0.001 (5)
753   664   1:normal   1:normal        *0.999  0.001 (5)
754   665   1:normal   1:normal        *0.999  0.001 (5)
755   666   1:normal   1:normal        *0.999  0.001 (5)
756   667   1:normal   1:normal        *0.999  0.001 (5)
757   668   1:normal   1:normal        *0.999  0.001 (5)
758   669   1:normal   1:normal        *0.999  0.001 (5)
759   670   1:normal   1:normal        *0.999  0.001 (5)
760   671   1:normal   1:normal        *0.999  0.001 (5)
761   672   1:normal   1:normal        *0.999  0.001 (5)
762   673   1:normal   1:normal        *0.999  0.001 (5)
763   674   1:normal   1:normal        *0.999  0.001 (5)
764   675   1:normal   1:normal        *0.999  0.001 (5)
765   676   1:normal   1:normal        *0.999  0.001 (5)
766   677   1:normal   1:normal        *0.999  0.001 (5)
767   678   1:normal   1:normal        *0.999  0.001 (5)
768   679   1:normal   1:normal        *0.999  0.001 (5)
769   680   1:normal   1:normal        *0.999  0.001 (5)
770   681   1:normal   1:normal        *0.999  0.001 (5)
771   682   1:normal   1:normal        *0.999  0.001 (5)
772   683   1:normal   1:normal        *0.999  0.001 (5)
773   684   1:normal   1:normal        *0.999  0.001 (5)
774   685   1:normal   1:normal        *0.999  0.001 (5)
775   686   1:normal   1:normal        *0.999  0.001 (5)
776   687   1:normal   1:normal        *0.999  0.001 (5)
777   688   1:normal   1:normal        *0.999  0.001 (5)
```

```
778    689   1:normal   1:normal         *0.999  0.001 (5)
779    690   1:normal   1:normal         *0.999  0.001 (5)
780    691   1:normal   1:normal         *0.999  0.001 (5)
781    692   1:normal   1:normal         *0.999  0.001 (5)
782    693   1:normal   1:normal         *0.999  0.001 (5)
783    694   1:normal   1:normal         *0.999  0.001 (5)
784    695   1:normal   1:normal         *0.999  0.001 (5)
785    696   1:normal   1:normal         *0.999  0.001 (5)
786    697   1:normal   1:normal         *0.999  0.001 (5)
787    698   1:normal   1:normal         *0.999  0.001 (5)
788    699   1:normal   1:normal         *0.999  0.001 (5)
789    700   1:normal   1:normal         *0.999  0.001 (5)
790    701   1:normal   1:normal         *0.999  0.001 (5)
791    702   1:normal   1:normal         *0.999  0.001 (5)
792    703   1:normal   1:normal         *0.999  0.001 (5)
793    704   1:normal   1:normal         *0.999  0.001 (5)
794    705   1:normal   1:normal         *0.999  0.001 (5)
795    706   1:normal   1:normal         *0.999  0.001 (5)
796    707   1:normal   1:normal         *0.999  0.001 (5)
797    708   1:normal   1:normal         *0.999  0.001 (5)
798    709   1:normal   1:normal         *0.999  0.001 (5)
799    710   1:normal   1:normal         *0.999  0.001 (5)
800    711   1:normal   1:normal         *0.999  0.001 (5)
801    712   1:normal   1:normal         *0.999  0.001 (5)
802    713   1:normal   1:normal         *0.999  0.001 (5)
803    714   1:normal   1:normal         *0.999  0.001 (5)
804    715   1:normal   1:normal         *0.999  0.001 (5)
805    716   1:normal   1:normal         *0.999  0.001 (5)
806    717   1:normal   1:normal         *0.999  0.001 (5)
807    718   1:normal   1:normal         *0.999  0.001 (5)
808    719   1:normal   1:normal         *0.999  0.001 (5)
809    720   1:normal   1:normal         *0.999  0.001 (5)
810    721   1:normal   1:normal         *0.999  0.001 (5)
811    722   1:normal   1:normal         *0.999  0.001 (5)
812    723   1:normal   1:normal         *1      0     (5)
```

```
813    724   1:normal   1:normal          *1      0     (5)
814    725   1:normal   1:normal          *0.999  0.001 (5)
815    726   1:normal   1:normal          *0.999  0.001 (5)
816    727   1:normal   1:normal          *0.999  0.001 (5)
817    728   1:normal   1:normal          *0.999  0.001 (5)
818    729   1:normal   1:normal          *0.999  0.001 (5)
819    730   1:normal   1:normal          *0.999  0.001 (5)
820    731   1:normal   1:normal          *0.999  0.001 (5)
821    732   1:normal   1:normal          *0.999  0.001 (5)
822    733   1:normal   1:normal          *0.999  0.001 (5)
823    734  2:anomalo  2:anomalo           0     *1     (5)
824    735  2:anomalo  2:anomalo           0     *1     (5)
825    736  2:anomalo  2:anomalo           0     *1     (5)
826    737  2:anomalo  2:anomalo           0     *1     (5)
827    738  2:anomalo  2:anomalo           0     *1     (5)
828    739  2:anomalo  2:anomalo           0     *1     (5)
829    740  2:anomalo  2:anomalo           0     *1     (5)
830    741  2:anomalo  2:anomalo           0     *1     (5)
831    742  2:anomalo  2:anomalo           0     *1     (5)
832    743  2:anomalo  2:anomalo           0     *1     (5)
833    744  2:anomalo  2:anomalo           0     *1     (5)
834    745   1:normal   1:normal          *0.999  0.001 (5)
835    746   1:normal   1:normal          *1      0     (5)
836    747   1:normal   1:normal          *1      0     (5)
837    748   1:normal   1:normal          *0.999  0.001 (5)
838    749   1:normal   1:normal          *1      0     (5)
839    750   1:normal   1:normal          *0.999  0.001 (5)
840    751  2:anomalo   1:normal      +   *0.999  0.001 (5)
841    752   1:normal   1:normal          *1      0     (5)
842    753   1:normal   1:normal          *1      0     (5)
843    754   1:normal   1:normal          *1      0     (5)
844    755   1:normal   1:normal          *1      0     (5)
845    756   1:normal   1:normal          *1      0     (5)
846    757   1:normal   1:normal          *1      0     (5)
847    758   1:normal   1:normal          *1      0     (5)
```

```
848   759   1:normal   1:normal        *1      0     (5)
849   760   1:normal   1:normal        *1      0     (5)
850   761   1:normal   1:normal        *1      0     (5)
851   762   1:normal   1:normal        *1      0     (5)
852   763   1:normal   1:normal        *0.999  0.001 (5)
853   764   1:normal   1:normal        *0.999  0.001 (5)
854   765   1:normal   1:normal        *0.999  0.001 (5)
855   766   1:normal   1:normal        *0.999  0.001 (5)
856   767   1:normal   1:normal        *0.999  0.001 (5)
857   768   1:normal   1:normal        *0.999  0.001 (5)
858   769   1:normal   1:normal        *0.999  0.001 (5)
859   770   1:normal   1:normal        *1      0     (5)
860   771   1:normal   1:normal        *1      0     (5)
861   772   1:normal   1:normal        *1      0     (5)
862   773   1:normal   1:normal        *1      0     (5)
863   774   1:normal   1:normal        *0.999  0.001 (5)
864   775   1:normal   1:normal        *0.999  0.001 (5)
865   776   1:normal   1:normal        *0.999  0.001 (5)
866   777   1:normal   1:normal        *0.999  0.001 (5)
867   778   1:normal   1:normal        *1      0     (5)
868   779   1:normal   1:normal        *1      0     (5)
869   780   1:normal   1:normal        *1      0     (5)
870   781   1:normal   1:normal        *1      0     (5)
871   782   1:normal   1:normal        *1      0     (5)
872   783   1:normal   1:normal        *1      0     (5)
873   784   1:normal   1:normal        *0.999  0.001 (5)
874   785   1:normal   1:normal        *0.999  0.001 (5)
875   786   1:normal   1:normal        *0.999  0.001 (5)
876   787   1:normal   1:normal        *1      0     (5)
877   788   1:normal   1:normal        *1      0     (5)
878   789   1:normal   1:normal        *1      0     (5)
879   790   1:normal   1:normal        *0.999  0.001 (5)
880   791   1:normal   1:normal        *1      0     (5)
881   792   1:normal   1:normal        *1      0     (5)
882   793   1:normal   1:normal        *1      0     (5)
```

```
883    794   1:normal   1:normal          *0.999  0.001 (5)
884    795   1:normal   1:normal          *0.999  0.001 (5)
885    796   1:normal   1:normal          *0.999  0.001 (5)
886    797   1:normal   1:normal          *0.999  0.001 (5)
887    798   1:normal   1:normal          *0.999  0.001 (5)
888    799   1:normal   1:normal          *0.999  0.001 (5)
889    800   1:normal   1:normal          *0.999  0.001 (5)
890    801   1:normal   1:normal          *0.999  0.001 (5)
891    802   1:normal   1:normal          *0.999  0.001 (5)
892    803   1:normal   1:normal          *0.999  0.001 (5)
893    804  2:anomalo  2:anomalo           0     *1      (5)
894    805   1:normal   1:normal          *1      0      (5)
895    806   1:normal   1:normal          *0.999  0.001 (5)
896    807  2:anomalo   1:normal      +   *0.999  0.001 (5)
897    808   1:normal   1:normal          *0.999  0.001 (5)
898    809   1:normal   1:normal          *0.999  0.001 (5)
899    810   1:normal   1:normal          *0.999  0.001 (5)
900    811   1:normal   1:normal          *0.999  0.001 (5)
901    812   1:normal   1:normal          *0.999  0.001 (5)
902    813   1:normal   1:normal          *0.999  0.001 (5)
903    814   1:normal   1:normal          *0.999  0.001 (5)
904    815   1:normal   1:normal          *0.999  0.001 (5)
905    816   1:normal   1:normal          *0.999  0.001 (5)
906    817   1:normal   1:normal          *0.999  0.001 (5)
907    818   1:normal   1:normal          *0.999  0.001 (5)
908    819   1:normal   1:normal          *0.999  0.001 (5)
909    820   1:normal   1:normal          *0.999  0.001 (5)
910    821   1:normal   1:normal          *0.999  0.001 (5)
911    822   1:normal   1:normal          *0.999  0.001 (5)
912    823   1:normal   1:normal          *1      0      (5)
913    824   1:normal   1:normal          *0.999  0.001 (5)
914    825   1:normal   1:normal          *0.999  0.001 (5)
915    826   1:normal   1:normal          *0.999  0.001 (5)
916    827   1:normal   1:normal          *0.999  0.001 (5)
917    828   1:normal   1:normal          *0.999  0.001 (5)
```

```
829   1:normal   1:normal        *0.999  0.001 (5)
830   1:normal   1:normal        *0.999  0.001 (5)
831   1:normal   1:normal        *0.999  0.001 (5)
832   1:normal   1:normal        *0.999  0.001 (5)
833   1:normal   1:normal        *0.999  0.001 (5)
834   1:normal   1:normal        *0.999  0.001 (5)
835   1:normal   1:normal        *0.999  0.001 (5)
836   1:normal   1:normal        *0.999  0.001 (5)
837   1:normal   1:normal        *0.999  0.001 (5)
838   1:normal   1:normal        *0.999  0.001 (5)
839   1:normal   1:normal        *0.999  0.001 (5)
840   1:normal   1:normal        *0.999  0.001 (5)
841   1:normal   1:normal        *0.999  0.001 (5)
842   1:normal   1:normal        *0.999  0.001 (5)
843   1:normal   1:normal        *0.999  0.001 (5)
844   1:normal   1:normal        *0.999  0.001 (5)
845   1:normal   1:normal        *0.999  0.001 (5)
846   1:normal   1:normal        *0.999  0.001 (5)
847   1:normal   1:normal        *0.999  0.001 (5)
848   1:normal   1:normal        *0.999  0.001 (5)
849   1:normal   1:normal        *0.999  0.001 (5)
850   1:normal   1:normal        *0.999  0.001 (5)
851   1:normal   1:normal        *0.999  0.001 (5)
852   1:normal   1:normal        *0.999  0.001 (5)
853   1:normal   1:normal        *0.999  0.001 (5)
854   1:normal   1:normal        *0.999  0.001 (5)
855   1:normal   1:normal        *0.999  0.001 (5)
856   1:normal   1:normal        *0.999  0.001 (5)
857   1:normal   1:normal        *0.999  0.001 (5)
858   1:normal   1:normal        *0.999  0.001 (5)
859   1:normal   1:normal        *0.999  0.001 (5)
860   1:normal   1:normal        *0.999  0.001 (5)
861   1:normal   1:normal        *0.999  0.001 (5)
862   1:normal   1:normal        *0.999  0.001 (5)
863   1:normal   1:normal        *0.999  0.001 (5)
```

```
953   864   1:normal   1:normal        *0.999  0.001 (5)
954   865   1:normal   1:normal        *0.999  0.001 (5)
955   866   1:normal   1:normal        *0.999  0.001 (5)
956   867   1:normal   1:normal        *0.999  0.001 (5)
957   868   1:normal   1:normal        *0.999  0.001 (5)
958   869   1:normal   1:normal        *0.999  0.001 (5)
959   870   1:normal   1:normal        *0.999  0.001 (5)
960   871   1:normal   1:normal        *0.999  0.001 (5)
961   872   1:normal   1:normal        *0.999  0.001 (5)
962   873   1:normal   1:normal        *0.999  0.001 (5)
963   874   1:normal   1:normal        *0.999  0.001 (5)
964   875   1:normal   1:normal        *0.999  0.001 (5)
965   876   1:normal   1:normal        *0.999  0.001 (5)
966   877   1:normal   1:normal        *0.999  0.001 (5)
967   878   1:normal   1:normal        *0.999  0.001 (5)
968   879   1:normal   1:normal        *0.999  0.001 (5)
969   880   1:normal   1:normal        *0.999  0.001 (5)
970   881   1:normal   1:normal        *0.999  0.001 (5)
971   882   1:normal   1:normal        *0.999  0.001 (5)
972   883   1:normal   1:normal        *0.999  0.001 (5)
973   884   1:normal   1:normal        *0.999  0.001 (5)
974   885   1:normal   1:normal        *0.999  0.001 (5)
975   886   1:normal   1:normal        *0.999  0.001 (5)
976   887   1:normal   1:normal        *0.999  0.001 (5)
977   888   1:normal   1:normal        *0.999  0.001 (5)
978   889   1:normal   1:normal        *0.999  0.001 (5)
979   890   1:normal   1:normal        *1      0     (5)
980   891   1:normal   1:normal        *0.999  0.001 (5)
981   892   1:normal   1:normal        *1      0     (5)
982   893   1:normal   1:normal        *1      0     (5)
983   894   1:normal   1:normal        *1      0     (5)
984   895   1:normal   1:normal        *1      0     (5)
985   896   1:normal   1:normal        *1      0     (5)
986   897   1:normal   1:normal        *1      0     (5)
987   898   1:normal   1:normal        *1      0     (5)
```

```
988   899   1:normal   1:normal        *1      0     (5)
989   900   1:normal   1:normal        *1      0     (5)
990   901   1:normal   1:normal        *1      0     (5)
991   902   1:normal   1:normal        *1      0     (5)
992   903   1:normal   1:normal        *1      0     (5)
993   904   1:normal   1:normal        *0.999  0.001 (5)
994   905   1:normal   1:normal        *0.999  0.001 (5)
995   906   1:normal   1:normal        *0.999  0.001 (5)
996   907   1:normal   1:normal        *0.999  0.001 (5)
997   908   1:normal   1:normal        *0.999  0.001 (5)
998   909   1:normal   1:normal        *0.999  0.001 (5)
999   910   1:normal   1:normal        *0.999  0.001 (5)
1000  911   1:normal   1:normal        *0.999  0.001 (5)
1001  912   1:normal   1:normal        *0.999  0.001 (5)
1002  913   1:normal   1:normal        *0.999  0.001 (5)
1003  914   1:normal   1:normal        *0.999  0.001 (5)
1004  915   1:normal   1:normal        *0.999  0.001 (5)
1005  916   1:normal   1:normal        *0.999  0.001 (5)
1006  917   1:normal   1:normal        *0.999  0.001 (5)
1007  918   1:normal   1:normal        *0.999  0.001 (5)
1008  919   1:normal   1:normal        *0.999  0.001 (5)
1009  920   1:normal   1:normal        *0.999  0.001 (5)
1010  921   1:normal   1:normal        *0.999  0.001 (5)
1011  922   1:normal   1:normal        *0.999  0.001 (5)
1012  923   1:normal   1:normal        *0.999  0.001 (5)
1013  924   1:normal   1:normal        *0.999  0.001 (5)
1014  925   1:normal   1:normal        *0.999  0.001 (5)
1015  926   1:normal   1:normal        *0.999  0.001 (5)
1016  927   1:normal   1:normal        *0.999  0.001 (5)
1017  928   1:normal   1:normal        *0.999  0.001 (5)
1018  929   1:normal   1:normal        *0.999  0.001 (5)
1019  930   1:normal   1:normal        *0.999  0.001 (5)
1020  931   1:normal   1:normal        *0.999  0.001 (5)
1021  932   1:normal   1:normal        *0.999  0.001 (5)
1022  933   1:normal   1:normal        *0.999  0.001 (5)
```

```
1023   934   1:normal   1:normal        *0.999  0.001 (5)
1024   935   1:normal   1:normal        *0.999  0.001 (5)
1025   936   1:normal   1:normal        *0.999  0.001 (5)
1026   937   1:normal   1:normal        *0.999  0.001 (5)
1027   938   1:normal   1:normal        *0.999  0.001 (5)
1028   939   1:normal   1:normal        *0.999  0.001 (5)
1029   940   1:normal   1:normal        *0.999  0.001 (5)
1030   941   1:normal   1:normal        *1      0     (5)
1031   942   1:normal   1:normal        *1      0     (5)
1032   943   1:normal   1:normal        *1      0     (5)
1033   944   1:normal   1:normal        *1      0     (5)
1034   945   1:normal   1:normal        *1      0     (5)
1035   946   1:normal   1:normal        *1      0     (5)
1036   947   1:normal   1:normal        *1      0     (5)
1037   948   1:normal   1:normal        *1      0     (5)
1038   949   1:normal   1:normal        *1      0     (5)
1039   950   1:normal   1:normal        *1      0     (5)
1040   951   1:normal   1:normal        *1      0     (5)
1041   952   1:normal   1:normal        *1      0     (5)
1042   953   1:normal   1:normal        *1      0     (5)
1043   954   1:normal   1:normal        *1      0     (5)
1044   955   1:normal   1:normal        *1      0     (5)
1045   956   1:normal   1:normal        *1      0     (5)
1046   957   1:normal   1:normal        *1      0     (5)
1047   958   1:normal   1:normal        *1      0     (5)
1048   959   1:normal   1:normal        *1      0     (5)
1049   960   1:normal   1:normal        *1      0     (5)
1050   961   1:normal   1:normal        *1      0     (5)
1051   962   1:normal   1:normal        *1      0     (5)
1052   963   1:normal   1:normal        *1      0     (5)
1053   964   1:normal   1:normal        *1      0     (5)
1054   965   1:normal   1:normal        *1      0     (5)
1055   966   1:normal   1:normal        *1      0     (5)
1056   967   1:normal   1:normal        *1      0     (5)
1057   968   1:normal   1:normal        *1      0     (5)
```

```
1058   969   1:normal   1:normal         *1      0     (5)
1059   970   1:normal   1:normal         *1      0     (5)
1060   971   1:normal   1:normal         *1      0     (5)
1061   972   1:normal   1:normal         *1      0     (5)
1062   973   1:normal   1:normal         *0.999  0.001 (5)
1063   974   1:normal   1:normal         *0.999  0.001 (5)
1064   975   1:normal   1:normal         *0.999  0.001 (5)
1065   976   1:normal   1:normal         *0.999  0.001 (5)
1066   977   1:normal   1:normal         *0.999  0.001 (5)
1067   978   1:normal   1:normal         *0.999  0.001 (5)
1068   979   1:normal   1:normal         *0.999  0.001 (5)
1069   980   1:normal   1:normal         *0.999  0.001 (5)
1070   981   1:normal   1:normal         *0.999  0.001 (5)
1071   982   1:normal   1:normal         *0.999  0.001 (5)
1072   983   1:normal   1:normal         *0.999  0.001 (5)
1073   984   1:normal   1:normal         *0.999  0.001 (5)
1074   985   1:normal   1:normal         *0.999  0.001 (5)
1075   986   1:normal   1:normal         *0.999  0.001 (5)
1076   987   1:normal   1:normal         *0.999  0.001 (5)
1077   988   1:normal   1:normal         *0.999  0.001 (5)
1078   989   1:normal   1:normal         *0.999  0.001 (5)
1079   990   1:normal   1:normal         *0.999  0.001 (5)
1080   991   1:normal   1:normal         *0.999  0.001 (5)
1081   992   1:normal   1:normal         *0.999  0.001 (5)
1082   993   1:normal   1:normal         *0.999  0.001 (5)
1083   994   1:normal   1:normal         *0.999  0.001 (5)
1084   995   1:normal   1:normal         *0.999  0.001 (5)
1085   996   1:normal   1:normal         *0.999  0.001 (5)
1086   997   1:normal   1:normal         *0.999  0.001 (5)
1087   998   1:normal   1:normal         *0.999  0.001 (5)
1088   999   1:normal   1:normal         *0.999  0.001 (5)
1089  1000   1:normal   1:normal         *0.999  0.001 (5)
1090  1001   1:normal   1:normal         *0.999  0.001 (5)
1091  1002   1:normal   1:normal         *0.999  0.001 (5)
1092  1003   1:normal   1:normal         *0.999  0.001 (5)
```

```
1093  1004   1:normal   1:normal         *0.999  0.001 (5)
1094  1005   1:normal   1:normal         *0.999  0.001 (5)
1095  1006   1:normal   1:normal         *0.999  0.001 (5)
1096  1007   1:normal   1:normal         *0.999  0.001 (5)
1097  1008   1:normal   1:normal         *0.999  0.001 (5)
1098  1009   1:normal   1:normal         *0.999  0.001 (5)
1099  1010   1:normal   1:normal         *0.999  0.001 (5)
1100  1011   1:normal   1:normal         *0.999  0.001 (5)
1101  1012   1:normal   1:normal         *0.999  0.001 (5)
1102  1013   1:normal   1:normal         *0.999  0.001 (5)
1103  1014   1:normal   1:normal         *0.999  0.001 (5)
1104  1015   1:normal   1:normal         *0.999  0.001 (5)
1105  1016   1:normal   1:normal         *0.999  0.001 (5)
1106  1017   1:normal   1:normal         *0.999  0.001 (5)
1107  1018   1:normal   1:normal         *0.999  0.001 (5)
1108  1019   1:normal   1:normal         *0.999  0.001 (5)
1109  1020   1:normal   1:normal         *0.999  0.001 (5)
1110  1021   1:normal   1:normal         *0.999  0.001 (5)
1111  1022   1:normal   1:normal         *0.999  0.001 (5)
1112  1023   1:normal   1:normal         *0.999  0.001 (5)
1113  1024   1:normal   1:normal         *0.999  0.001 (5)
1114  1025   1:normal   1:normal         *0.999  0.001 (5)
1115  1026   1:normal   1:normal         *0.999  0.001 (5)
1116  1027   1:normal   1:normal         *0.999  0.001 (5)
1117  1028   1:normal   1:normal         *0.999  0.001 (5)
1118  1029   1:normal   1:normal         *0.999  0.001 (5)
1119  1030   1:normal   1:normal         *0.999  0.001 (5)
1120  1031   1:normal   1:normal         *0.999  0.001 (5)
1121  1032   1:normal   1:normal         *0.999  0.001 (5)
1122  1033   1:normal   1:normal         *0.999  0.001 (5)
1123  1034   1:normal   1:normal         *0.999  0.001 (5)
1124  1035   1:normal   1:normal         *0.999  0.001 (5)
1125  1036  2:anomalo  2:anomalo          0     *1     (5)
1126  1037  2:anomalo  2:anomalo          0     *1     (5)
1127  1038  2:anomalo  2:anomalo          0     *1     (5)
```

```
1039  2:anomalo  2:anomalo          0     *1     (5)
1040  2:anomalo  2:anomalo          0     *1     (5)
1041   1:normal   1:normal        *0.989  0.011 (5)
1042   1:normal   1:normal        *1      0     (5)
1043   1:normal   1:normal        *1      0     (5)
1044   1:normal   1:normal        *1      0     (5)
1045   1:normal   1:normal        *1      0     (5)
1046   1:normal   1:normal        *1      0     (5)
1047   1:normal   1:normal        *0.999  0.001 (5)
1048   1:normal   1:normal        *0.999  0.001 (5)
1049   1:normal   1:normal        *0.999  0.001 (5)
1050   1:normal   1:normal        *1      0     (5)
1051   1:normal   1:normal        *1      0     (5)
1052   1:normal   1:normal        *1      0     (5)
1053   1:normal   1:normal        *1      0     (5)
1054   1:normal   1:normal        *1      0     (5)
1055   1:normal   1:normal        *1      0     (5)
1056   1:normal   1:normal        *0.999  0.001 (5)
1057   1:normal   1:normal        *0.999  0.001 (5)
1058   1:normal   1:normal        *1      0     (5)
1059   1:normal   1:normal        *1      0     (5)
1060   1:normal   1:normal        *1      0     (5)
1061   1:normal   1:normal        *1      0     (5)
1062   1:normal   1:normal        *0.999  0.001 (5)
1063   1:normal   1:normal        *1      0     (5)
1064   1:normal   1:normal        *1      0     (5)
1065   1:normal   1:normal        *0.999  0.001 (5)
1066   1:normal   1:normal        *1      0     (5)
1067   1:normal   1:normal        *0.986  0.014 (5)
1068   1:normal   1:normal        *0.999  0.001 (5)
1069   1:normal   1:normal        *1      0     (5)
1070   1:normal   1:normal        *1      0     (5)
1071   1:normal   1:normal        *1      0     (5)
1072   1:normal   1:normal        *0.999  0.001 (5)
1073   1:normal   1:normal        *0.999  0.001 (5)
```

```
1074   1:normal   1:normal         *1      0     (5)
1075   1:normal   1:normal         *1      0     (5)
1076   1:normal   1:normal         *1      0     (5)
1077   1:normal   1:normal         *1      0     (5)
1078   1:normal   1:normal         *0.999  0.001 (5)
1079   1:normal   1:normal         *0.999  0.001 (5)
1080   1:normal   1:normal         *0.999  0.001 (5)
1081   1:normal   1:normal         *0.999  0.001 (5)
1082   1:normal   1:normal         *0.999  0.001 (5)
1083  2:anomalo  2:anomalo          0     *1     (5)
1084  2:anomalo  2:anomalo          0     *1     (5)
1085   1:normal   1:normal         *1      0     (5)
1086   1:normal   1:normal         *1      0     (5)
1087   1:normal   1:normal         *1      0     (5)
1088   1:normal   1:normal         *1      0     (5)
1089   1:normal   1:normal         *1      0     (5)
1090   1:normal   1:normal         *1      0     (5)
1091   1:normal   1:normal         *0.999  0.001 (5)
1092   1:normal   1:normal         *1      0     (5)
1093   1:normal   1:normal         *1      0     (5)
1094   1:normal   1:normal         *0.999  0.001 (5)
1095   1:normal   1:normal         *0.999  0.001 (5)
1096   1:normal   1:normal         *1      0     (5)
1097   1:normal   1:normal         *1      0     (5)
1098   1:normal   1:normal         *1      0     (5)
1099   1:normal   1:normal         *1      0     (5)
1100  2:anomalo  2:anomalo          0     *1     (5)
1101  2:anomalo  2:anomalo          0     *1     (5)
1102   1:normal   1:normal         *0.999  0.001 (5)
1103   1:normal   1:normal         *1      0     (5)
1104   1:normal   1:normal         *0.999  0.001 (5)
1105   1:normal   1:normal         *1      0     (5)
1106   1:normal   1:normal         *0.999  0.001 (5)
1107   1:normal   1:normal         *1      0     (5)
1108   1:normal   1:normal         *0.999  0.001 (5)
```

```
1198   1109   1:normal   1:normal         *0.999  0.001 (5)
1199   1110   1:normal   1:normal         *1      0     (5)
1200   1111   1:normal   1:normal         *0.999  0.001 (5)
1201   1112   1:normal   1:normal         *1      0     (5)
1202   1113   1:normal   1:normal         *1      0     (5)
1203   1114   1:normal   1:normal         *0.999  0.001 (5)
1204   1115   1:normal   1:normal         *0.999  0.001 (5)
1205   1116   1:normal   1:normal         *1      0     (5)
1206   1117   1:normal   1:normal         *0.999  0.001 (5)
1207   1118   1:normal   1:normal         *1      0     (5)
1208   1119   1:normal   1:normal         *0.999  0.001 (5)
1209   1120   1:normal   1:normal         *1      0     (5)
1210   1121  2:anomalo  2:anomalo          0     *1     (5)
1211   1122  2:anomalo  2:anomalo          0     *1     (5)
1212   1123   1:normal   1:normal         *0.999  0.001 (5)
1213   1124   1:normal   1:normal         *1      0     (5)
1214   1125   1:normal   1:normal         *0.999  0.001 (5)
1215   1126   1:normal   1:normal         *1      0     (5)
1216   1127   1:normal   1:normal         *0.999  0.001 (5)
1217   1128   1:normal   1:normal         *1      0     (5)
1218   1129   1:normal   1:normal         *0.999  0.001 (5)
1219   1130   1:normal   1:normal         *1      0     (5)
1220   1131   1:normal   1:normal         *0.999  0.001 (5)
1221   1132   1:normal   1:normal         *1      0     (5)
1222   1133   1:normal   1:normal         *1      0     (5)
1223   1134   1:normal   1:normal         *0.999  0.001 (5)
1224   1135   1:normal   1:normal         *1      0     (5)
1225   1136   1:normal   1:normal         *0.999  0.001 (5)
1226   1137   1:normal   1:normal         *1      0     (5)
1227   1138  2:anomalo  2:anomalo          0     *1     (5)
1228   1139  2:anomalo  2:anomalo          0     *1     (5)
1229   1140   1:normal   1:normal         *1      0     (5)
1230   1141   1:normal   1:normal         *1      0     (5)
1231   1142   1:normal   1:normal         *0.999  0.001 (5)
1232   1143   1:normal   1:normal         *1      0     (5)
```

```
1233   1144   1:normal   1:normal         *0.999  0.001 (5)
1234   1145   1:normal   1:normal         *1      0     (5)
1235   1146   1:normal   1:normal         *1      0     (5)
1236   1147  2:anomalo  2:anomalo          0     *1     (5)
1237   1148  2:anomalo  2:anomalo          0     *1     (5)
1238   1149   1:normal   1:normal         *0.999  0.001 (5)
1239   1150   1:normal   1:normal         *0.999  0.001 (5)
1240   1151   1:normal   1:normal         *1      0     (5)
1241   1152   1:normal   1:normal         *1      0     (5)
1242   1153  2:anomalo  2:anomalo          0     *1     (5)
1243   1154  2:anomalo  2:anomalo          0     *1     (5)
1244   1155   1:normal   1:normal         *0.999  0.001 (5)
1245   1156   1:normal   1:normal         *1      0     (5)
1246   1157   1:normal   1:normal         *1      0     (5)
1247   1158   1:normal   1:normal         *1      0     (5)
1248   1159   1:normal   1:normal         *0.999  0.001 (5)
1249   1160   1:normal   1:normal         *0.999  0.001 (5)
1250   1161  2:anomalo  2:anomalo          0     *1     (5)
1251   1162  2:anomalo  2:anomalo          0     *1     (5)
1252   1163   1:normal   1:normal         *0.999  0.001 (5)
1253   1164   1:normal   1:normal         *1      0     (5)
1254   1165   1:normal   1:normal         *0.999  0.001 (5)
1255   1166   1:normal   1:normal         *1      0     (5)
1256   1167  2:anomalo  2:anomalo          0     *1     (5)
1257   1168  2:anomalo  2:anomalo          0     *1     (5)
1258   1169   1:normal   1:normal         *0.999  0.001 (5)
1259   1170   1:normal   1:normal         *1      0     (5)
1260   1171  2:anomalo  2:anomalo          0     *1     (5)
1261   1172  2:anomalo  2:anomalo          0     *1     (5)
1262   1173  2:anomalo  2:anomalo          0     *1     (5)
1263   1174  2:anomalo  2:anomalo          0     *1     (5)
1264   1175   1:normal   1:normal         *1      0     (5)
1265   1176  2:anomalo  2:anomalo          0     *1     (5)
1266   1177  2:anomalo  2:anomalo          0     *1     (5)
1267   1178   1:normal   1:normal         *1      0     (5)
```

```
1179  2:anomalo  2:anomalo          0     *1     (5)
1180  2:anomalo  2:anomalo          0     *1     (5)
1181   1:normal   1:normal         *0.999  0.001 (5)
1182   1:normal   1:normal         *0.999  0.001 (5)
1183   1:normal   1:normal         *1      0     (5)
1184   1:normal   1:normal         *1      0     (5)
1185  2:anomalo  2:anomalo          0     *1     (5)
1186  2:anomalo  2:anomalo          0     *1     (5)
1187   1:normal   1:normal         *0.999  0.001 (5)
1188  2:anomalo  2:anomalo          0     *1     (5)
1189  2:anomalo  2:anomalo          0     *1     (5)
1190  2:anomalo  2:anomalo          0     *1     (5)
1191  2:anomalo  2:anomalo          0     *1     (5)
1192  2:anomalo  2:anomalo          0     *1     (5)
1193  2:anomalo  2:anomalo          0     *1     (5)
1194  2:anomalo  2:anomalo          0     *1     (5)
1195  2:anomalo  2:anomalo          0     *1     (5)
1196   1:normal   1:normal         *0.999  0.001 (5)
1197   1:normal   1:normal         *1      0     (5)
1198   1:normal   1:normal         *0.999  0.001 (5)
1199   1:normal   1:normal         *1      0     (5)
1200   1:normal   1:normal         *1      0     (5)
1201   1:normal   1:normal         *0.999  0.001 (5)
1202   1:normal   1:normal         *0.999  0.001 (5)
1203   1:normal   1:normal         *0.999  0.001 (5)
1204   1:normal   1:normal         *1      0     (5)
1205   1:normal   1:normal         *0.999  0.001 (5)
1206   1:normal   1:normal         *0.999  0.001 (5)
1207   1:normal   1:normal         *0.999  0.001 (5)
1208   1:normal   1:normal         *1      0     (5)
1209   1:normal   1:normal         *1      0     (5)
1210   1:normal   1:normal         *0.999  0.001 (5)
1211   1:normal   1:normal         *0.999  0.001 (5)
1212   1:normal   1:normal         *0.999  0.001 (5)
1213   1:normal   1:normal         *1      0     (5)
```

```
1214   1:normal   1:normal         *1      0     (5)
1215   1:normal   1:normal         *0.999  0.001 (5)
1216   1:normal   1:normal         *0.999  0.001 (5)
1217   1:normal   1:normal         *1      0     (5)
1218   1:normal   1:normal         *1      0     (5)
1219  2:anomalo  2:anomalo          0     *1     (5)
1220  2:anomalo  2:anomalo          0     *1     (5)
1221  2:anomalo  2:anomalo          0     *1     (5)
1222   1:normal   1:normal         *0.999  0.001 (5)
1223   1:normal   1:normal         *0.999  0.001 (5)
1224   1:normal   1:normal         *0.999  0.001 (5)
1225   1:normal   1:normal         *0.999  0.001 (5)
1226   1:normal   1:normal         *1      0     (5)
1227   1:normal   1:normal         *0.999  0.001 (5)
1228   1:normal   1:normal         *1      0     (5)
1229   1:normal   1:normal         *0.999  0.001 (5)
1230   1:normal   1:normal         *1      0     (5)
1231   1:normal   1:normal         *0.999  0.001 (5)
1232   1:normal   1:normal         *1      0     (5)
1233   1:normal   1:normal         *0.999  0.001 (5)
1234   1:normal   1:normal         *1      0     (5)
1235   1:normal   1:normal         *0.999  0.001 (5)
1236   1:normal   1:normal         *0.999  0.001 (5)
1237   1:normal   1:normal         *1      0     (5)
1238   1:normal   1:normal         *0.999  0.001 (5)
1239   1:normal   1:normal         *1      0     (5)
1240   1:normal   1:normal         *1      0     (5)
1241   1:normal   1:normal         *0.999  0.001 (5)
1242   1:normal   1:normal         *1      0     (5)
1243   1:normal   1:normal         *0.999  0.001 (5)
1244   1:normal   1:normal         *0.999  0.001 (5)
1245   1:normal   1:normal         *0.999  0.001 (5)
1246   1:normal   1:normal         *1      0     (5)
1247   1:normal   1:normal         *1      0     (5)
1248   1:normal   1:normal         *0.999  0.001 (5)
```

```
1249   1:normal   1:normal        *1      0     (5)
1250   1:normal   1:normal        *1      0     (5)
1251   1:normal   1:normal        *1      0     (5)
1252   1:normal   1:normal        *1      0     (5)
1253   1:normal   1:normal        *1      0     (5)
1254   1:normal   1:normal        *0.999  0.001 (5)
1255   1:normal   1:normal        *1      0     (5)
1256   1:normal   1:normal        *0.999  0.001 (5)
1257   1:normal   1:normal        *0.999  0.001 (5)
1258   1:normal   1:normal        *0.999  0.001 (5)
1259   1:normal   1:normal        *0.999  0.001 (5)
1260   1:normal   1:normal        *0.999  0.001 (5)
1261   1:normal   1:normal        *1      0     (5)
1262   1:normal   1:normal        *1      0     (5)
1263   1:normal   1:normal        *1      0     (5)
1264   1:normal   1:normal        *1      0     (5)
1265   1:normal   1:normal        *0.999  0.001 (5)
1266   1:normal   1:normal        *1      0     (5)
1267   1:normal   1:normal        *0.999  0.001 (5)
1268   1:normal   1:normal        *1      0     (5)
1269   1:normal   1:normal        *0.999  0.001 (5)
1270   1:normal   1:normal        *1      0     (5)
1271   1:normal   1:normal        *1      0     (5)
1272   1:normal   1:normal        *0.999  0.001 (5)
1273   1:normal   1:normal        *0.999  0.001 (5)
1274   1:normal   1:normal        *1      0     (5)
1275   1:normal   1:normal        *1      0     (5)
1276   1:normal   1:normal        *0.999  0.001 (5)
1277   1:normal   1:normal        *0.999  0.001 (5)
1278   1:normal   1:normal        *0.999  0.001 (5)
1279   1:normal   1:normal        *0.999  0.001 (5)
1280   1:normal   1:normal        *1      0     (5)
1281   1:normal   1:normal        *1      0     (5)
1282   1:normal   1:normal        *0.999  0.001 (5)
1283   1:normal   1:normal        *1      0     (5)
```

```
1284   1:normal   1:normal         *1      0     (5)
1285   1:normal   1:normal         *0.999  0.001 (5)
1286   1:normal   1:normal         *0.999  0.001 (5)
1287   1:normal   1:normal         *1      0     (5)
1288   1:normal   1:normal         *0.999  0.001 (5)
1289   1:normal   1:normal         *0.999  0.001 (5)
1290   1:normal   1:normal         *0.999  0.001 (5)
1291   1:normal   1:normal         *0.999  0.001 (5)
1292   1:normal   1:normal         *1      0     (5)
1293   1:normal   1:normal         *1      0     (5)
1294   1:normal   1:normal         *1      0     (5)
1295   1:normal   1:normal         *1      0     (5)
1296   1:normal   1:normal         *0.999  0.001 (5)
1297  2:anomalo  2:anomalo          0     *1     (5)
1298  2:anomalo  2:anomalo          0     *1     (5)
1299  2:anomalo  2:anomalo          0     *1     (5)
1300  2:anomalo  2:anomalo          0     *1     (5)
1301  2:anomalo  2:anomalo          0     *1     (5)
1302  2:anomalo  2:anomalo          0     *1     (5)
1303  2:anomalo  2:anomalo          0     *1     (5)
1304  2:anomalo  2:anomalo          0     *1     (5)
1305  2:anomalo  2:anomalo          0     *1     (5)
1306  2:anomalo  2:anomalo          0     *1     (5)
1307   1:normal   1:normal         *1      0     (5)
1308  2:anomalo  2:anomalo          0     *1     (5)
1309  2:anomalo  2:anomalo          0     *1     (5)
1310  2:anomalo  2:anomalo          0     *1     (5)
1311   1:normal   1:normal         *1      0     (5)
1312   1:normal   1:normal         *1      0     (5)
1313  2:anomalo  2:anomalo          0     *1     (5)
1314  2:anomalo  2:anomalo          0     *1     (5)
1315  2:anomalo  2:anomalo          0     *1     (5)
1316   1:normal   1:normal         *1      0     (5)
1317  2:anomalo  2:anomalo          0     *1     (5)
1318  2:anomalo  2:anomalo          0     *1     (5)
```

```
1408   1319  2:anomalo  2:anomalo          0     *1     (5)
1409   1320  2:anomalo  2:anomalo          0     *1     (5)
1410   1321  2:anomalo  2:anomalo          0     *1     (5)
1411   1322  2:anomalo  2:anomalo          0     *1     (5)
1412   1323  2:anomalo  2:anomalo          0     *1     (5)
1413   1324  2:anomalo  2:anomalo          0     *1     (5)
1414   1325  2:anomalo  2:anomalo          0     *1     (5)
1415   1326  2:anomalo  2:anomalo          0     *1     (5)
1416   1327  2:anomalo  2:anomalo          0     *1     (5)
1417   1328  2:anomalo  2:anomalo          0     *1     (5)
1418   1329  2:anomalo  2:anomalo          0     *1     (5)
1419   1330  2:anomalo  2:anomalo          0     *1     (5)
1420   1331  2:anomalo  2:anomalo          0     *1     (5)
1421   1332  2:anomalo  2:anomalo          0     *1     (5)
1422   1333  2:anomalo  2:anomalo          0     *1     (5)
1423   1334  2:anomalo  2:anomalo          0     *1     (5)
1424   1335  2:anomalo  2:anomalo          0     *1     (5)
1425   1336  2:anomalo  2:anomalo          0     *1     (5)
1426   1337  2:anomalo  2:anomalo          0     *1     (5)
1427   1338  2:anomalo  2:anomalo          0     *1     (5)
1428   1339  2:anomalo  2:anomalo          0     *1     (5)
1429   1340  2:anomalo  2:anomalo          0     *1     (5)
1430   1341   1:normal   1:normal         *1      0     (5)
1431   1342  2:anomalo  2:anomalo          0     *1     (5)
1432   1343  2:anomalo  2:anomalo          0     *1     (5)
1433   1344  2:anomalo  2:anomalo          0     *1     (5)
1434   1345  2:anomalo  2:anomalo          0     *1     (5)
1435   1346   1:normal   1:normal         *1      0     (5)
1436   1347  2:anomalo  2:anomalo          0     *1     (5)
1437   1348  2:anomalo  2:anomalo          0     *1     (5)
1438   1349  2:anomalo  2:anomalo          0     *1     (5)
1439   1350  2:anomalo  2:anomalo          0     *1     (5)
1440   1351   1:normal   1:normal       *0.999  0.001  (5)
1441   1352  2:anomalo  2:anomalo          0     *1     (5)
1442   1353  2:anomalo  2:anomalo          0     *1     (5)
```

```
1443   1354  2:anomalo  2:anomalo          0     *1     (5)
1444   1355  2:anomalo  2:anomalo          0     *1     (5)
1445   1356   1:normal   1:normal         *1      0     (5)
1446   1357  2:anomalo  2:anomalo          0     *1     (5)
1447   1358  2:anomalo  2:anomalo          0     *1     (5)
1448   1359  2:anomalo  2:anomalo          0     *1     (5)
1449   1360  2:anomalo  2:anomalo          0     *1     (5)
1450   1361  2:anomalo  2:anomalo          0     *1     (5)
1451   1362  2:anomalo  2:anomalo          0     *1     (5)
1452   1363  2:anomalo  2:anomalo          0     *1     (5)
1453   1364  2:anomalo  2:anomalo          0     *1     (5)
1454   1365  2:anomalo  2:anomalo          0     *1     (5)
1455   1366  2:anomalo  2:anomalo          0     *1     (5)
1456   1367  2:anomalo  2:anomalo          0     *1     (5)
1457   1368  2:anomalo  2:anomalo          0     *1     (5)
1458   1369  2:anomalo  2:anomalo          0     *1     (5)
1459   1370  2:anomalo  2:anomalo          0     *1     (5)
1460   1371  2:anomalo  2:anomalo          0     *1     (5)
1461   1372  2:anomalo  2:anomalo          0     *1     (5)
1462   1373   1:normal   1:normal         *1      0     (5)
1463   1374  2:anomalo  2:anomalo          0     *1     (5)
1464   1375  2:anomalo  2:anomalo          0     *1     (5)
1465   1376  2:anomalo  2:anomalo          0     *1     (5)
1466   1377  2:anomalo  2:anomalo          0     *1     (5)
1467   1378  2:anomalo  2:anomalo          0     *1     (5)
1468   1379  2:anomalo  2:anomalo          0     *1     (5)
1469   1380  2:anomalo  2:anomalo          0     *1     (5)
1470   1381  2:anomalo  2:anomalo          0     *1     (5)
1471   1382  2:anomalo  2:anomalo          0     *1     (5)
1472   1383  2:anomalo  2:anomalo          0     *1     (5)
1473   1384  2:anomalo  2:anomalo          0     *1     (5)
1474   1385  2:anomalo  2:anomalo          0     *1     (5)
1475   1386  2:anomalo  2:anomalo          0     *1     (5)
1476   1387  2:anomalo  2:anomalo          0     *1     (5)
1477   1388  2:anomalo  2:anomalo          0     *1     (5)
```

```
  1389  2:anomalo  2:anomalo          0     *1     (5)
  1390  2:anomalo  2:anomalo          0     *1     (5)
  1391  2:anomalo  2:anomalo          0     *1     (5)
  1392  2:anomalo  2:anomalo          0     *1     (5)
  1393  2:anomalo  2:anomalo          0     *1     (5)
  1394  2:anomalo  2:anomalo          0     *1     (5)
  1395  2:anomalo  2:anomalo          0     *1     (5)
  1396  2:anomalo  2:anomalo          0     *1     (5)
  1397  2:anomalo  2:anomalo          0     *1     (5)
  1398   1:normal   1:normal         *1      0     (5)
  1399  2:anomalo  2:anomalo          0     *1     (5)
  1400  2:anomalo  2:anomalo          0     *1     (5)
  1401  2:anomalo  2:anomalo          0     *1     (5)
  1402  2:anomalo  2:anomalo          0     *1     (5)
  1403  2:anomalo  2:anomalo          0     *1     (5)

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances        1395               99.4298 %
Incorrectly Classified Instances         8                0.5702 %
Kappa statistic                          0.9718
Mean absolute error                      0.0067
Root mean squared error                  0.0752
Relative absolute error                  5.6215 %
Root relative squared error             22.0589 %
Total Number of Instances             1403

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.049      0.994     1         0.997      0.978    normal
                 0.951     0          1         0.951     0.975      0.978    anomalo
Weighted Avg.    0.994     0.043      0.994     0.994     0.994      0.978

```

```
=== Confusion Matrix ===

    a    b   <-- classified as
 1239    0 |    a = normal
    8  156 |    b = anomalo
```

A.5 Log de treino com algoritmo Naive Bayes

```
=== Run information ===

Scheme:weka.classifiers.bayes.NaiveBayes
Relation:     outcsv
Instances:    89015
Attributes:   17
              EFT
              HS
              TOS
              PS
              ID
              FL
              FR
              PT
              SPORT
              DPORT
              UDPS
              TCPS
              TCPHS
              IDP
              ACK
              DF
              TP
Test mode:evaluate on training data

```

```
=== Classifier model (full training set) ===

Naive Bayes Classifier

                         Class
Attribute               normal         anomalo
                           (1)             (0)
================================================
EFT
  mean                      1               1
  std. dev.            0.0017          0.0017
  weight sum            88755             260
  precision              0.01            0.01

HS
  mean                 5.0106               5
  std. dev.            0.1667          0.1667
  weight sum            88755             260
  precision                 1               1

TOS
  mean                 0.2012               0
  std. dev.                 8               8
  weight sum            88755             260
  precision                48              48

PS
  mean               561.4503        155.8518
  std. dev.          817.4476        273.0939
  weight sum            88755             260
  precision            24.308          24.308

ID
  mean             25366.0054       5397.8135
  std. dev.        20535.7502      12497.3143
```

```
  weight sum              88755             260
  precision               1.382           1.382

FL
  mean                  12.3743          2.9224
  std. dev.              7.1206          6.2214
  weight sum              88755             260
  precision             16.1667         16.1667

FR
  mean           1661481646.656  689701845.7729
  std. dev.     1433892911.9446  1283034801.641
  weight sum              88755             260
  precision          111021.705      111021.705

PT
  UDP                   18892.0           192.0
  IP                      993.0             1.0
  TCP                   68873.0            70.0
  [total]               88758.0           263.0

SPORT
  mean                5587.4853       11145.3341
  std. dev.          12424.9152       19810.2665
  weight sum              88755             260
  precision             29.6073         29.6073

DPORT
  mean                 4097.338        12817.925
  std. dev.           9342.1844       13557.8643
  weight sum              88755             260
  precision             50.1371         50.1371

UDPS
  mean                  57.2125         25.4812
```

```
  std. dev.            128.1536         38.5091
  weight sum              88755             260
  precision              5.7861          5.7861

TCPS
  mean                  455.44          94.4015
  std. dev.             838.471         274.8204
  weight sum              88755             260
  precision             25.1996         25.1996

TCPHS
  mean                   3.8936          1.6258
  std. dev.               2.254          2.9215
  weight sum              88755             260
  precision              1.5714          1.5714

IDP
  mean                   7.5441         13.1692
  std. dev.              4.4326          4.7097
  weight sum              88755             260
  precision              5.3333          5.3333

ACK
  mean          1660819802.1265  358147440.3284
  std. dev.     1451365756.1976   944438646.234
  weight sum              88755             260
  precision          173823.904      173823.904

DF
  mean                   0.3481          0.3538
  std. dev.              0.4764          0.4782
  weight sum              88755             260
  precision                   1               1
```

```
131
132 Time taken to build model: 0.19 seconds
133
134 === Evaluation on training set ===
135 === Summary ===
136
137 Correctly Classified Instances       83810               94.1527 %
138 Incorrectly Classified Instances      5205                5.8473 %
139 Kappa statistic                          0.0655
140 Mean absolute error                      0.0577
141 Root mean squared error                  0.2286
142 Relative absolute error                988.1676 %
143 Root relative squared error            423.5495 %
144 Total Number of Instances             89015
145
146 === Detailed Accuracy By Class ===
147
148                TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
149                  0.942     0.238      0.999     0.942     0.97       0.913    normal
150                  0.762     0.058      0.037     0.762     0.071      0.913    anomalo
151 Weighted Avg.    0.942     0.238      0.996     0.942     0.967      0.913
152
153 === Confusion Matrix ===
154
155      a     b   <-- classified as
156  83612  5143 |     a = normal
157     62   198 |     b = anomalo
```

A.6 Log de teste com algoritmo Naive Bayes

```
1 === Run information ===
2
3 Scheme:weka.classifiers.bayes.NaiveBayes
```

```
Relation:     outcsv
Instances:    89015
Attributes:   17
              EFT
              HS
              TOS
              PS
              ID
              FL
              FR
              PT
              SPORT
              DPORT
              UDPS
              TCPS
              TCPHS
              IDP
              ACK
              DF
              TP
Test mode:user supplied test set: size unknown (reading incrementally)

=== Classifier model (full training set) ===

Naive Bayes Classifier

                         Class
Attribute               normal         anomalo
                           (1)             (0)
================================================
EFT
  mean                      1               1
  std. dev.            0.0017          0.0017
  weight sum            88755             260
  precision              0.01            0.01
```

```
39
40 HS
41   mean                  5.0106              5
42   std. dev.             0.1667         0.1667
43   weight sum             88755            260
44   precision                  1              1
45
46 TOS
47   mean                  0.2012              0
48   std. dev.                  8              8
49   weight sum             88755            260
50   precision                 48             48
51
52 PS
53   mean                561.4503        155.8518
54   std. dev.           817.4476        273.0939
55   weight sum             88755            260
56   precision             24.308         24.308
57
58 ID
59   mean              25366.0054       5397.8135
60   std. dev.         20535.7502      12497.3143
61   weight sum             88755            260
62   precision              1.382          1.382
63
64 FL
65   mean                 12.3743          2.9224
66   std. dev.             7.1206          6.2214
67   weight sum             88755            260
68   precision            16.1667         16.1667
69
70 FR
71   mean          1661481646.656  689701845.7729
72   std. dev.     1433892911.9446  1283034801.641
73   weight sum             88755            260
```

```
  precision          111021.705      111021.705

PT
  UDP                  18892.0           192.0
  IP                     993.0             1.0
  TCP                  68873.0            70.0
  [total]              88758.0           263.0

SPORT
  mean                5587.4853      11145.3341
  std. dev.          12424.9152      19810.2665
  weight sum             88755             260
  precision            29.6073         29.6073

DPORT
  mean                4097.338       12817.925
  std. dev.          9342.1844      13557.8643
  weight sum             88755             260
  precision            50.1371         50.1371

UDPS
  mean                 57.2125         25.4812
  std. dev.           128.1536         38.5091
  weight sum             88755             260
  precision             5.7861          5.7861

TCPS
  mean                  455.44         94.4015
  std. dev.             838.471        274.8204
  weight sum             88755             260
  precision            25.1996         25.1996

TCPHS
  mean                  3.8936          1.6258
  std. dev.              2.254          2.9215
```

```
  weight sum              88755             260
  precision              1.5714          1.5714

IDP
  mean                   7.5441         13.1692
  std. dev.              4.4326          4.7097
  weight sum              88755             260
  precision              5.3333          5.3333

ACK
  mean          1660819802.1265  358147440.3284
  std. dev.     1451365756.1976   944438646.234
  weight sum              88755             260
  precision          173823.904      173823.904

DF
  mean                   0.3481          0.3538
  std. dev.              0.4764          0.4782
  weight sum              88755             260
  precision                   1               1



Time taken to build model: 0.19 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances        1380               98.3607 %
Incorrectly Classified Instances        23                1.6393 %
Kappa statistic                          0.9195
Mean absolute error                      0.0194
Root mean squared error                  0.1275
Relative absolute error                 16.2813 %
Root relative squared error             37.4089 %
```

```
Total Number of Instances             1403

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.993     0.085      0.989     0.993     0.991      0.954    normal
                 0.915     0.007      0.943     0.915     0.929      0.955    anomalo
Weighted Avg.    0.984     0.076      0.983     0.984     0.983      0.954

=== Confusion Matrix ===

    a    b   <-- classified as
 1230    9 |    a = normal
   14  150 |    b = anomalo
```

A.7 Log de treino com algoritmo SVM

```
=== Run information ===

Scheme:weka.classifiers.functions.LibSVM -S 0 -K 2 -D 3 -G 0.0 -R 0.0 -N 0.5 -M 40.0 -
    C 1.0 -E 0.001 -P 0.1 -seed 1
Relation:     outcsv
Instances:    89015
Attributes:   17
              EFT
              HS
              TOS
              PS
              ID
              FL
              FR
              PT
              SPORT
```

```
16                DPORT
17                UDPS
18                TCPS
19                TCPHS
20                IDP
21                ACK
22                DF
23                TP
24 Test mode:evaluate on training data
25
26 === Classifier model (full training set) ===
27
28 LibSVM wrapper, original code by Yasser EL-Manzalawy (= WLSVM)
29
30 Time taken to build model: 5219.86 seconds
31
32 === Evaluation on training set ===
33 === Summary ===
34
35 Correctly Classified Instances       88998               99.9809 %
36 Incorrectly Classified Instances        17                0.0191 %
37 Kappa statistic                          0.9661
38 Mean absolute error                      0.0002
39 Root mean squared error                  0.0138
40 Relative absolute error                  3.2726 %
41 Root relative squared error             25.6078 %
42 Total Number of Instances            89015
43
44 === Detailed Accuracy By Class ===
45
46                TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
47                  1         0.065      1         1         1          0.967    normal
48                  0.935     0          1         0.935     0.966      0.967    anomalo
49 Weighted Avg.    1         0.065      1         1         1          0.967
50
```

```
=== Confusion Matrix ===

     a     b   <-- classified as
 88755     0 |     a = normal
    17   243 |     b = anomalo
```

A.8 Log de Teste com algoritmo SVM

```
=== Run information ===

Scheme:weka.classifiers.functions.LibSVM -S 0 -K 2 -D 3 -G 0.0 -R 0.0 -N 0.5 -M 40.0 -
    C 1.0 -E 0.001 -P 0.1 -seed 1
Relation:     outcsv
Instances:    89015
Attributes:   17
              EFT
              HS
              TOS
              PS
              ID
              FL
              FR
              PT
              SPORT
              DPORT
              UDPS
              TCPS
              TCPHS
              IDP
              ACK
              DF
              TP
Test mode:user supplied test set: size unknown (reading incrementally)
```

```
25
26 === Classifier model (full training set) ===
27
28 LibSVM wrapper, original code by Yasser EL-Manzalawy (= WLSVM)
29
30 Time taken to build model: 5134.39 seconds
31
32 === Evaluation on test set ===
33 === Summary ===
34
35 Correctly Classified Instances        1403               100      %
36 Incorrectly Classified Instances         0                 0      %
37 Kappa statistic                          1
38 Mean absolute error                      0
39 Root mean squared error                  0
40 Relative absolute error                  0      %
41 Root relative squared error              0      %
42 Total Number of Instances             1403
43
44 === Detailed Accuracy By Class ===
45
46                TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
47                  1         0          1         1         1          1        normal
48                  1         0          1         1         1          1        anomalo
49 Weighted Avg.    1         0          1         1         1          1
50
51 === Confusion Matrix ===
52
53    a    b   <-- classified as
54  1239    0 |    a = normal
55    0  164 |    b = anomalo
```