CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Igor Henrique Martini

GERAÇÃO DE *DATASETS* COM ATAQUES CRIPTOGRAFADOS PARA IDS

Santa Cruz do Sul

2017

Igor Henrique Martini

GERAÇÃO DE *DATASETS* COM ATAQUES CRIPTOGRAFADOS PARA IDS

Trabalho de Conclusão II apresentado ao Curso de Ciência da Computação da Universidade de Santa Cruz do Sul - UNISC, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Charles Varlei Neu

Santa Cruz do Sul

2017

AGRADECIMENTOS

A elaboração deste trabalho não teria sido possível sem a colaboração, estímulo e empenho de diversas pessoas. Gostaria, por este fato, de expressar toda a minha gratidão e apreço a todos aqueles que, direta ou indiretamente, contribuíram para que esta tarefa se tornasse uma realidade. A todos quero manifestar os meus sinceros agradecimentos.

Agradeço primeiramente a Deus, por ter me dado saúde e força para superar as dificuldades. Também agradeço a minha mãe Norma Sueli Maurer Martini, que não mediu esforços para que eu chegasse até esta etapa de minha vida. Faço um agradecimento especial a minha avó Maria Maurer, pelo incentivo em todos os momentos e todo o apoio que me deu possibilitando a realização desse sonho. Agradeço a minha namorada Julia Fernanda Tatsch, pela paciência, pelo incentivo nas horas difíceis, de desânimo e cansaço.

Ao meu orientador, Prof. Me. Charles Varlei Neu, pelo suporte no pouco tempo que lhe coube, pelas suas correções e incentivos.

A todos, meus eternos e sinceros agradecimentos.

RESUMO

Com a necessidade de dispositivos que estabeleçam comunicação entre si para a troca de informações, redes de computadores estão sendo amplamente utilizadas. Porém, alguns usuários promovem a má utilização desta rede com o intuito de prejudicar o funcionamento da mesma, utilizam recursos disponíveis de forma ilegal ou tentam adquirir informações sem autorização. A criptografia que era vista como uma proteção tornou-se uma tecnologia que facilita a entrada de ataques na rede, visto que atacantes estão utilizando este recurso para mascarar seus ataques. A maioria dos dispositivos de segurança, como por exemplo, o IDS (Sistema de Detecção de Intrusão), é incapaz de detectar este tipo de intrusão, a menos que esse tráfego seja previamente decifrado. Para validação de um IDS faz-se necessário o uso de uma base de dados, também conhecida como *dataset* para IDS. Os *datasets* são conjuntos de dados gerados através da coleta de pacotes na rede. Este trabalho apresenta a criação de *datasets* com pacotes cifrados e aplica criptografia sobre *datasets* existentes, provendo bases com ataques criptografados. Estes *datasets* são importantes para testar métodos de detecção de intrusão a fim de identificar atividades maliciosas em fluxos cifrados.

Palavras-chave: Segurança de redes**,** Sistema de Detecção de Intrusão, Dataset, Criptografia, IDS

ABSTRACT

Due to the growing need of devices that communicate amongst themselves for exchange of information, computer networks are being widely used. However, there are users who misuse these networks with the purpose of exploiting them, illegally using available resources or trying to get unauthorized information. The encryption that was seen as a protection has become a technology that facilitates the entry of attacks on the network, as attackers are using this feature to mask their attacks. Most security devices, such as the IDS (Intrusion Detection System), are unable to detect this type of intrusion, unless this traffic has previously been deciphered. To validate an IDS, it is necessary to use a database, also known as IDS datasets. These are data sets generated by collecting packages on the network. This paper proposes the creation of datasets containing encrypted packages, applying encryption algorithms on existing datasets. These datasets are important for testing intrusion detection methods in order to identify malicious activity in encrypted traffic.

Keywords: Network Security**,** Intrusion Detection System, Dataset, Encryption**,** IDS

LISTA DE ABREVIATURAS

CSV	Comma-separated values
DIDS	Distributed Intrusion Detection System
DDoS	Distributed Denial of Service
DoS	Denial of Service
DPI	Deep Packet Inspection
FTP	File Transfer Protocol
HIDS	Host Intrusion Detection System
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IMAP	Internet Message Access Protocol
NIDS	Network Intrusion Detection System
PCAP	Packet Capture
POP	Post Office Protocol
SMTP	Simple Mail Transfer Protocol
SO	Sistema Operacional
SSH	Secure Shell
TCP/IP	Transmission Control Protocol / Internet Protocol
XOR	Exclusive OR

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Com o rápido desenvolvimento da tecnologia baseada na Internet e a subordinação de negócios críticos aos sistemas de informação, novos domínios de aplicação em redes de computadores vêm surgindo (STALLINGS, 2014). Na medida em que as redes se evoluem, existe também o aumento das vulnerabilidades que podem comprometê-las. A segurança da informação neste contexto torna-se essencial para garantir a proteção dos dados que trafegam nestas redes.

A todo o momento novos tipos de ataque surgem, tornando imprescindível a criação de mecanismos de defesa mais sofisticados. Os ataques podem corromper os dados de uma aplicação e, dependendo de seu tipo e intensidade, podem até mesmo levar o sistema a um colapso. Um exemplo de ataque que ocorreu recentemente foi o Ransomware Wanna Cry (SYMANTEC, 2017), um vírus projetado para bloquear arquivos de usuários até o pagamento de resgate. Os *hackers* criptografaram os arquivos de diversos usuários pelo mundo todo e exigiram que os mesmos pagassem uma taxa para descriptografá-los. Com isso, várias medidas vêm sendo criadas para garantir a segurança contra ataques. A criptografia e autenticação são os mecanismos de prevenção da primeira linha de defesa em uma rede, garantindo alguns princípios de segurança como confidencialidade e integridade (STALLINGS, 2014). Através da criptografia é possível transformar uma mensagem em um emaranhado de caracteres deixando um texto simples impossível de ser compreendido. Apesar de ser uma técnica de alta segurança é possível decifrá-la e recuperar a mensagem após ela ter sido criptografada. No entanto, é necessário que a pessoa seja autorizada a obter essa informação com a chave de decriptação.

Porém, quando estas medidas não são suficientes para lidar com todos os tipos de ataque, faz-se necessário um segundo mecanismo de segurança, os Sistemas de Detecção de Intrusão (*Intrusion Detection System* - IDS) (DEBAR, 2000). De modo geral, um IDS analisa o tráfego de rede tentando reconhecer um ataque ou uma invasão para alertar o administrador ou automaticamente disparar contramedidas. As técnicas utilizadas para detectar intrusões normalmente são classificadas em dois tipos: assinatura e anomalia (FAGUNDES, 2016). A detecção por assinatura compara o tráfego com uma base de dados de ataques conhecidos (assinaturas), enquanto a detecção por anomalia compara os dados coletados com registros de atividades consideradas normais no sistema. Um desvio significativo da normalidade é considerado uma ameaça. Ambas as abordagens possuem várias desvantagens. A detecção por

assinatura exige atualização frequente dos registros para garantir uma boa detecção, enquanto a detecção por anomalia sofre de uma alta taxa de falsos positivos (FAGUNDES, 2016).

As bases de dados com registros de ataques desempenham um importante papel na validação de um IDS, e podem ser chamadas também de *Datasets* para IDS. A qualidade dos dados permite não só avaliar a habilidade do método proposto na detecção de comportamento intrusivo, mas também mostra a potencial eficácia do IDS no ambiente operacional (TAVALLAEE *et al*., 2010).

O IDS reconhece apenas os ataques descritos em sua base de dados e qualquer ataque com diferentes características, como ataques cifrados farão com que a intrusão passe despercebida. Portanto, o problema encontrando reside no fato de que os *datasets* tradicionais existentes, não contém ataques criptografados, entretanto, diversos pesquisadores estão trabalhando para resolver esta dificuldade.

Embora a criptografia sempre tenha sido vista como um meio de proteção, hoje ela também se tornou uma tecnologia que facilita a entrada de atacantes na rede interna. *Hackes* utilizam criptografia para mascarar seus ataques usufruindo de brechas nos métodos de IDS, que não conseguem decifrar os pacotes cifrados por não possuir uma base de dados com estas características, ou seja, com ataques criptografados (NEU *et al*., 2016).

Tendo sido apresentados os conceitos envolvendo a proposta desta pesquisa, apresenta-se como objetivo a criação de *datasets* contendo ataques cifrados, aplicando algoritmos de criptografia simétrica em alguns campos dos *datasets*. Estas bases serão importantes para testar métodos de detecção de intrusão com fluxo criptografado, visto que não foram encontrados *datasets* com tais características.

Para criar os *datasets* cifrados, desenvolveu-se uma ferramenta para abrir os pacotes de um *dataset* e cifrar sua área de dados (*payload),* mantendo o cabeçalho original. Para testar a ferramenta utilizou-se as bases de dados ISCX IDS 2012 e hydra_telnet. Foi desempenhada uma análise sobre estes *datasets*, onde foi determinado um fluxo de pacotes com ataques para aplicar a criptografia. A validação dos resultados foi feita através da ferramenta Wireshark, onde foi possível presenciar a aplicação da cifragem no *payload* dos pacotes.

Este trabalho de conclusão está organizado da seguinte maneira: o Capítulo 2 apresenta os conceitos necessários para a correta compreensão do mesmo. O Capítulo 3 refere-se ao Estado da arte e trabalhos relacionados a detecção de intrusão criptografada. O trabalho desenvolvido é apresentado no Capitulo 4. O Capítulo 5 exibe os testes realizados e os resultados obtidos. E finalmente, a conclusão e sugestão de trabalhos futuros são apresentados no Capítulo 6.

2 FUNDAMENTAÇÃO TEÓRICA

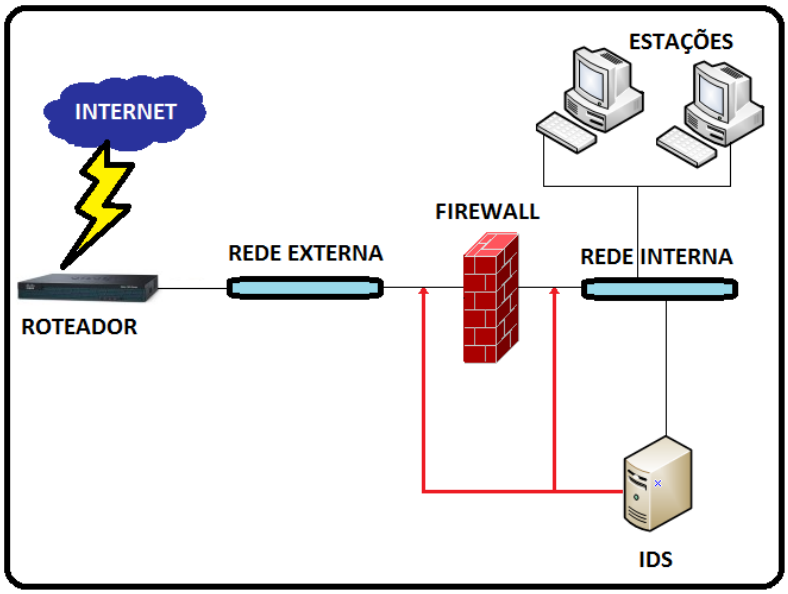
Nesta seção, é apresentado o referencial teórico com os conceitos necessários para a realização deste trabalho, visto que são fundamentais para a compreensão e desenvolvimento do mesmo.

2.1 IDS - Sistemas de Detecção de Intrusão

Detecção de Intrusão é o processo que monitora e analisa eventos sistematizados de computação ou redes que tem como objetivo descobrir sinais de prováveis incidentes que podem comprometer a integridade, disponibilidade de recursos e a confidencialidade do sistema computacional (FAGUNDES, 2016). Utilizando os métodos de detecção de intrusão podemos armazenar informações sobre determinados tipos de ataques já existentes e anteceder possíveis tentativas de ataques à rede. Estas informações armazenadas podem ser utilizadas para construção de uma base informativa e também para tomada de decisões.

Podemos definir IDS como um sistema de configurações e regras com a finalidade de disparar alertas quando detecta pacotes que fazem parte de um provável ataque. Um IDS também pode ser definido como uma ferramenta inteligente capaz de detectar tentativas de invasão em tempo real. Na Figura 1 podemos analisar o funcionamento de um IDS, onde temos três placas de rede que serão o scanner das redes externa e interna (setas vermelhas), e ele é ligado em um *Switch* que filtra todas as entradas e saídas da rede para verificar a existência de algum protocolo ou requisição maliciosa (GODOI, 2007).

Figura 1: Funcionamento de um IDS



Fonte: (Adaptado de: GODOI, 2007)

Os métodos de detecção dos IDS dividem-se em duas categorias básicas: detecção em assinaturas de intrusão e detecção de anomalia (estatística).

- **Detecção baseada em assinaturas:** baseiam-se em um conjunto de condições que caracterizam a manifestação direta de atividades de intrusão em termos de cabeçalhos de pacotes e conteúdo útil (*payload).* O método baseado em assinaturas sempre foi o mais utilizado em termos de NIDS. Este método, tem como referência a sua base de dados de assinaturas de ataques e quando uma ou mais dessas assinaturas é detectada, é disparado um alarme e o evento é registrado para futuras investigações. O ponto forte da detecção de intrusão baseada em assinatura, está fortemente relacionada com a qualidade e atualização da base de dados de assinaturas.
- **Detecção baseada em anomalias:** Um IDS é capaz de identificar um ataque quando a ocorrência de algum evento é diferente de algum padrão considerado normal, por exemplo, alguma aplicação realizando uma tentativa de acesso não autorizado a algum recurso do sistema (SNORT, 2015).

Os sistemas IDS são agrupados em três categorias: NIDS, HIDS e DIDS.

2.1.1 NIDS - *Network Intrusion Detection System*

NIDS são sistemas que monitoram o tráfego da rede procurando por pacotes maliciosos. Este sistema é composto por um módulo de captura de pacotes chamado *sniffer*. O módulo opera configurando a placa de rede utilizada para monitoramento em modo promíscuo. Uma placa de rede operando em modo normal recebe todos os pacotes até detectar o endereço de destino do mesmo (FAGUNDES, 2016).

Caso o pacote não esteja endereçado ao próprio computador, ou não seja um pacote *broadcast* (endereçado a todos os nós da rede) ou *multicast* (endereçado a um grupo de computadores) ao qual aquele computador pertence, o mesmo será rejeitado. Em modo promíscuo, a placa de rede fará a captura de todos os pacotes observados independentemente do endereço de destino (FAGUNDES, 2016).

2.1.2 HIDS - *Host Intrusion Detection System*

O HIDS é responsável por monitorar a atividade de um dispositivo, isto é, ele busca identificar ataques ou eventos suspeitos que comprometam a segurança deste dispositivo, analisando apenas as informações transmitidas pela rede destinadas a este dispositivo. A característica principal e que distingue esta categoria é que a visão está restrita apenas ao "*host*" ou computador no qual ele está instalado e operando (FAGUNDES, 2016).

São compostos por sistemas especialistas que monitoram a utilização de recursos como disco, memória, processador e chamadas de funções do SO, com o intuito de procurar por padrões que determinem um ataque ou por desvios relevantes em relação ao perfil de uso considerado normal e regular. Uma grande vantagem do HIDS é a capacidade de detectar mudanças especificas nos arquivos e no sistema operacional do *host*. É possível monitorar tamanho de arquivos a fim de garantir que arquivos do sistema não foram maliciosamente modificados (FAGUNDES, 2016).

2.1.3 DIDS - *Distributed Intrusion Detection System*

DIDS é uma combinação de sensores NIDS e HIDS com regras específicas relativamente à sua localização, são distribuídos em uma rede, e os alertas são encaminhados a um sistema central. *Logs* de ataques são gerados nos sensores e enviados para o sistema central, onde eles serão armazenados em um banco de dados central (FAGUNDES, 2016).

Novas assinaturas são adquiridas da Internet para a estação de gerenciamento assim que disponíveis, e essas assinaturas podem ser enviadas para os sensores de acordo com suas necessidades bases. O que distingue essa categoria é que ela possui uma estação de gerenciamento no qual os sensores se reportam ou os mesmos podem trocar informações entre si (FAGUNDES, 2016).

2.2 Datasets

As bases de dados com registros de ataques desempenham um importante papel na validação de um IDS, e podem ser chamadas também de *datasets* para IDS. A qualidade dos dados permite não só avaliar a habilidade do método proposto na detecção de comportamento intrusivo, mas também mostra a potencial eficácia do IDS no ambiente operacional. A grande dificuldade em se estudar cenários de utilização de detecção de intrusão é a necessidade de se possuir uma base de dados para treinamento (NEU *et al*., 2016).

Atualmente, diversos *datasets* são encontrados para testes de IDS, porém eles possuem diferentes características. O ***dataset labeled*** possui a informação (*label*) dos pacotes que podem ou não ser maliciosos, identificando que tipos de ataques representam, além dos pacotes normais. Pode-se considerar um ***dataset unlabeled*** aquele que não traz a rotulagem (*label)* dos pacotes. Temos também o ***dataset flow-based*** para detecção de intrusão baseada em fluxo (SPEROTTO *et al.*, 2009), este tem como característica os fluxos (*flows),* que são conjuntos de pacotes trafegando na rede durante um determinado intervalo de tempo e possuem um conjunto de propriedades comuns. O ***dataset packet-based*** possui a carga útil dos pacotes, por exemplo, as informações que foram transmitidas, e este é o principal atributo deste tipo de *dataset.*

Na sequência, serão apresentados alguns exemplos de *Datasets* encontrados na literatura. Os *Datasets* escolhidos para utilizar neste trabalho de conclusão foram ISCX IDS 2012 e hydra_telnet, pelo fato dos mesmos serem mais atuais e possuírem a carga útil dos dados.

2.2.1 KDDCUP99

O KDDCUP99 foi construído com base nos dados capturados pelo DARPA'S98, programa de avaliação de IDS (LIPPMANN, 2000). Ele possui 4GB de arquivo comprimido de dados brutos de conexão TCP, tendo sete semanas de tráfego de rede, que pode ser transformado em cerca de 5 milhões de registros de conexões cada um com 100bytes. Duas semanas de teste tem aproximadamente dois milhões de registros de conexão.

Os dados capturados consistiam de todos os pacotes IP, TCP, UDP, ICMP capturados nesta rede em formato *tcpdump* (TCPDUMP, 2016). Em meados de 1999, os organizadores de uma competição internacional em algoritmos de mineração de dados KDDCUP escolheram esta base de dados e utilizaram o problema de detecção de intrusão como tema de seu concurso anual. Entretanto, decidiram realizar uma etapa de pré-processamento na base de dados, reduzindo em uma base de dados de registros de conexões. Cada sequência de pacotes entre um mesmo par de endereços IP e portas de origem e destino específico, dentro de um intervalo de tempo pré-estabelecido tiveram suas principais características extraídas e concentradas em um único registro de conexão com 41 campos. O programa *tcptrace* (TCPTRACE, 2016) realiza esta consolidação de diversos pacotes em um arquivo capturado com *tcpdump* em registros de conexão individuais. Todos os registros de conexões receberam ainda um campo identificador classificando-a como uma conexão normal ou um ataque.

Atributos básicos de uma conexão TCP/IP formam o primeiro grupo de campos gerados. Foram denominadas características intrínsecas do TCP/IP e estão apresentadas na Tabela 1.

Tabela 1: Características intrínsecas de conexões TCP/IP

Nome	**Descrição**
Duration	Duração em segundos da conexão
Protocol_type	Tipo de protocolo usado na conexão, ex: tcp, udp, entre outros.
Service	Serviço de rede sendo utilizado, ex: http, telnet, ftp, entre outros.
Src_bytes	Número de bytes enviados da fonte para o destino
Dst_bytes	Número de bytes enviados do destino para a fonte
Flag	Status da conexão (normal ou erro)
Land	1 se conexão é de/para o mesmo host, 0 caso contrário
Wrong_fragment	Número de fragmentos com erro
Urgent	Número de pacotes com flag urgente habilitado

Fonte: (KDDCUP, 2016)

Atributos da conexão dependente de conhecimento especialista também foram extraídas da base de pacotes TCP/IP e estas transformadas em alguns campos nos registros de conexão. Estes atributos foram obtidos através de análise de informações obtidas na área de dados dos pacotes TCP, UDP e IP. Nesta área, encontra-se geralmente, cabeçalhos de aplicações de nível

superior como Telnet, ftp, http, rlogin. Na Tabela 2 são apresentadas as características de conexão por conhecimento especialista.

Tabela 2: Características de conexão por conhecimento especialista

Nome	**Descrição**
Hot	Número de indicadores chaves (“hot”).
Num_failed_logins	Tentativas de login sem sucesso
Logged_in	1 se login efetuado com sucesso; 0 caso contrário
Num_compromised	Número de condições de “comprometimento”
Root_shell	1 se shell root foi obtido; 0 caso contrário
Su_attempted	1 se comando “su root” foi tentado; 0 caso contrário
Num_root	Número de acessos como root
Num_file_creations	Número de operações de criação de arquivos
Num_shells	Número de “shells prompts” obtidos
Num_access_files	Número de operações em arquivos de controle de acesso
Num_outbound_cmds	Número de comandos externos em uma sessão ftp.
Is_hot_login	1 se o login pertence a lista “hot”; 0 caso contrário
Is_guest_login	1 se o login usou a conta guest; 0 caso contrário

Fonte: (KDDCUP, 2016)

Os demais campos foram gerados aplicando um intervalo de 2 segundos nos pacotes capturados. Esta aplicação torna-se importante para a detecção de técnicas de intrusão e ataques das classes de negação de serviço e reconhecimento. Na Tabela 3 são apresentadas as características temporais do *dataset*.

Tabela 3: Características temporais - janela de 2 segundos

Nome	**Descrição**
Count	Número de conexões iguais a esta para este mesmo "host" nos últimos 2 segundos.
Srv_count	Número de conexões para o mesmo serviço que o usado nesta conexão nos últimos 2 segundos
Serror_rate	% de conexões que possuem erro SYN.
Srv_serror_rate	% de conexões que possuem erro SYN para este serviço.
Rerror_rate	% de conexões que possuem erro REJ.
Srv_rerror_rate	% de conexões que possuem erro REJ para este serviço.
Same_srv_rate	% de conexões para um mesmo serviço.
Diff_srv_rate	% de conexões para serviços diferentes
Srv_diff_host_rate	% de conexões deste mesmo serviço para hosts diferentes
Dst_host_count	Número de conexões com mesmo host de destino que esta
Dst_host_srv_count	Número de conexões com mesmo host de destino e mesmo serviço que esta
Dst_host_same_srv_count	% de conexões com o mesmo host de destino e mesmo serviço que esta
Dst_host_diff_srv_rate	% de conexões com o mesmo host de destino e serviços diferentes que esta
Dst_host_same_src_port_rate	% de conexões com o mesmo host de destino e a mesma porta de origem que a conexão atual
Dst_host_srv_diff_host_rate	% de conexões para o mesmo serviço vindo de diferentes hosts
Dst_host_serror_rate	% de conexões para o mesmo host que o da conexão atual e que possui um erro S0.
Dst_host_srv_serror_rate	% de conexões para o mesmo host e serviço que o da conexão atual e que possui um erro S0.
Dst_host_rerror_rate	% de conexões para o mesmo host que apresentem flag RST.
Dst_host_srv_rerror_rate	% de conexões para o mesmo host e serviço da conexão atual que apresentem flag RST.

Fonte: (KDDCUP, 2016)

Cada ataque simulado entra em uma das quatro categorias, ilustradas na Tabela 4. As classes de ataque no KDDCUP99 podem ser distribuídas em 5 classes principais, uma normal e quatro classes de intrusão: DoS (*Denial of Service*), R2L (*Remote to Local),* U2R (*User to Root)* e *Probing*.

Tabela 4: Categorias de Ataque

Categoria do ataque	**Ataques**
DoS (Ataques de Negação de Serviço)	back, land, neptune, pod, smurf, teardrop
U2R (Ataques de Usuário para Administrador)	buffer_overflow, perl, loadmodule, rootkit
R2L (Ataques de Remoto para Local)	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
Probing (Ataques de Reconhecimento)	ipsweep, nmap, portsweep, satan

Fonte: (KDDCUP, 2016)

- Ataques de Negação de Serviço (DoS): tem como objetivo sobrecarregar algum recurso computacional, interrompendo atividades legais como simplismente navegar na Internet. Por exemplo, ataques como *Neptune* e *smurf* abusam de uma característica específica do serviço atacado, outros como *teardrop* e *pod-ping of death* criam pacotes com formatos errados, o que dificulta a reconstrução do pacote, enquanto o *back* utiliza erros de determinado serviço de rede (KENDALL, 1999);

- Ataques de Usuário para Administrador (U2R): neste tipo de ataque, o invasor consegue entrar no sistema como usuário comum e então explora falhas para conseguir privilégios de administrador. O ataque mais comum desta categoria é o *overflow,* que ocorre quando um programa copia muitos dados em um *buffer* estático sem verificação, permitindo ao usuário inserir mais dados do que os necessários, podendo então executar códigos arbitrários (KENDALL, 1999);

- Ataques de Remoto para Local (R2L): ocorre quando um invasor tem a habilidade de enviar pacotes para uma máquina através da rede, mas não possui uma conta nesta máquina, mas mesmo assim consegue explorar alguma falha para ganhar acesso local como usuário da máquina. Há várias formas de efetuar estes ataques, o *ftp-write* pode explorar por segurança de políticas fracas, *imap* pode utilizar *buffer overflow,* ou ainda pode utilizar engenharia social para infiltrar *trojans* (Xlock) e obter senhas de acesso (KENDALL, 1999);

- Ataques de Reconhecimento (Probing): ocorre quando o invasor faz uma tentativa de reunir informações sobre uma determinada rede com o intuito de burlar os controles de segurança. O invasor que possui um bom mapa de uma rede, pode obter informações sobre pontos mais fracos da mesma. Alguns aplicativos como *satan*, podem ser utilizados para varrer uma grande área de rede procurando por vulnerabilidades conhecidas. Esta técnica é geralmente usada na mineração de dados através das ferramentas *saint*, *portsweep*, *mscan* e *nmap* (KENDALL, 1999).

Existem dois conjuntos de dados, treino e teste, e eles não apresentam a mesma distribuição e probabilidade. Além disso, os dados de testes contêm registros não disponíveis nos dados de treino o que faz a tarefa mais realista. Muitos especialistas em segurança acreditam que novos ataques são variações de ataques conhecidos e que as características desses são suficientes para capturar novas variações (DE CAMPOS, 2012).

2.2.2 NSL-KDD

A base de dados NSL-KDD tem origem no conjunto de dados pertencentes ao KDDCUP99, sendo este uma versão da base inicial de dados criada pelo MIT Lincoln Labs, para o Programa de Avaliação de Detecção de Intrusão DARPA de 1998. Preocupados com a qualidade dos dados contidos no KDDCUP99, (TAVALLAEE *et al*., 2009) realizaram estudos e argumentaram sobre problemas na coleta dos dados, como a sobrecarga da ferramenta de coleta (*tcpdump*) quando havia alto tráfego e a diferença dos dados fornecidos com os dados costumeiramente coletados em uma rede real, e sobre a falta de explicações sobre definições específicas dos ataques, pois uma sondagem (*probing*), por exemplo, não é necessariamente um ataque até que ultrapasse um limite. Também levantaram problemas argumentados em outros artigos, sobre diversas críticas a base KDD, como por exemplo, a quantidade desproporcional de ataques nas bases de dados.

Para melhorar a qualidade dos dados, uma nova base foi proposta, o NSL-KDD (NSL-KDD, 2011), que retirou os dados redundantes e distribuiu de outra forma os dados do KDDCUP99, separando os dados em três conjuntos:

1. KDDTrain+: possui a base de treino;
2. KDDTest+: possui a base de testes;
3. KDDTest^{-21}: possui os dados que não foram bem classificados pelos algoritmos utilizados na criação da base.

Podemos citar as seguintes vantagens da base NSL-KDD em relação ao conjunto de dados original KDDCUP99:

- Não inclui registros redundantes;

- Não há registros duplicados;

- O número de registros selecionados de cada grupo *difficultylevel* é inversamente proporcional ao percentual de registros no conjunto de dados original KDD. Como resultado, as taxas de classificação dos métodos de aprendizado de máquina distinta variam em uma escala mais larga, o que torna mais eficiente para ter uma avaliação precisa das diferentes técnicas de aprendizagem;

- O número de registros nos dados de treinamento e teste são razoaveis, o que o torna acessível para executar os experimentos sobre o conjunto completo sem a necessidade de selecionar aleatoriamente uma pequena porção de dados. Consequentemente, os resultados da avaliação de diferentes trabalhos de pesquisa são consistentes e comparáveis.

Além desses, foi criado um conjunto com 20% dos dados de treino e aplicados diferentes métodos de aprendizagem fornecidos pelo WEKA (WEKA, 2011) na base original KDDCUP e em suas bases de testes.

O trabalho de (SILVA, 2005) desenvolve métodos de detecção de intrusão com técnicas de Inteligência Artificial como Algoritmos Genéticos, Redes Neurais Artificiais e Lógica Nebulosa, criando assim um SIH que realizou uma análise detalhada e melhorias das taxas de detecção com a base de dados NSL-KDD, além de fazer comparações com os resultados obtidos em Tavallaee (TAVALLAEE, 2009).

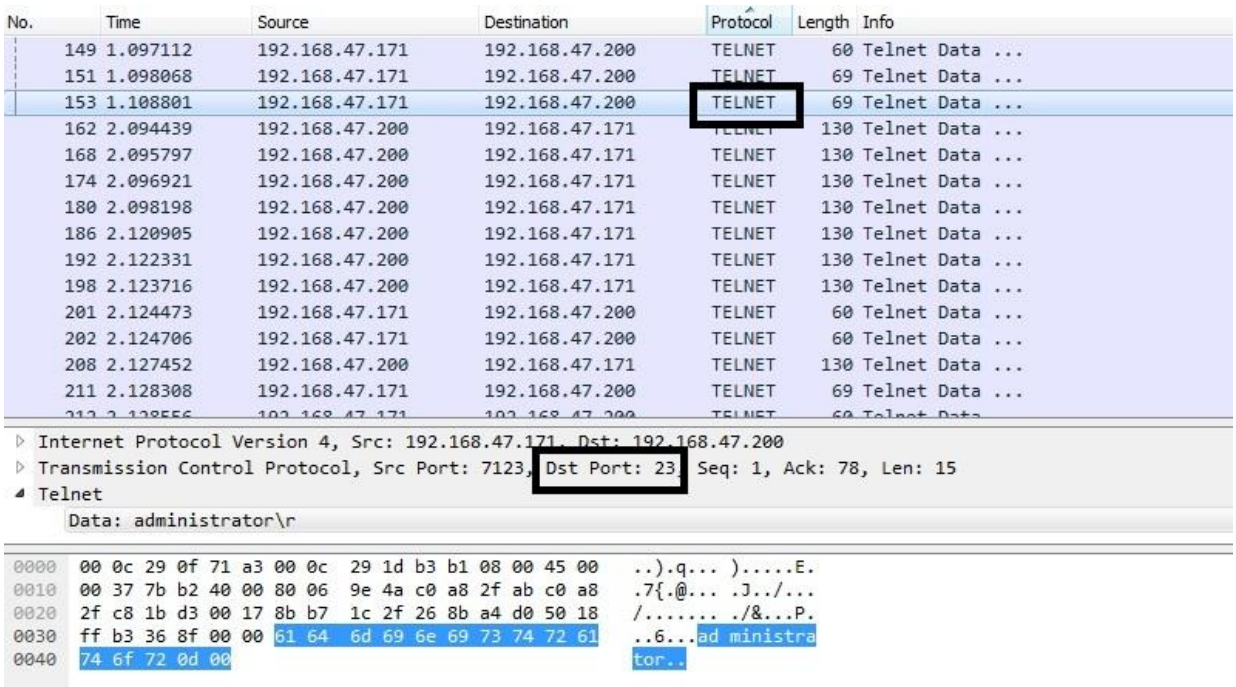
2.2.3 HYDRA_TELNET

Oliveira (GARCIA, 2016) criou um *dataset* contendo ataques de força bruta, que consiste em consecutivas tentativas de verificação de senhas sobre um serviço em um determinado alvo, a qual se utiliza um banco de palavras para verificação dos dados (THC, 2017). Esta ferramenta usa palavras dessa base e faz uma série de verificações com cada palavra até ser equivalente a real senha do host alvo. A ferramenta Hydra foi desenvolvida por Van Hauser e David Maciejak. Este *dataset* é o resultado de testes feitos utilizando a ferramenta Hydra para injetar ataques em uma rede SDN.

O protocolo utilizado para injetar os ataques foi o Telnet. A Figura 2 mostra o tráfego do ataque direcionado para a porta 23 do serviço de Telnet, onde os arquivos estão organizados

em sequência caracterizando a tentativa de intrusão, assim permitindo a análise de cada ataque em direção ao serviço específico (GARCIA, 2016).

Figura 2: Tráfego gerado pelo ataque no Wireshark



Fonte: (GARCIA, 2016)

Optou-se por utilizar o *dataset* hydra_telnet neste trabalho pois o mesmo traz consigo a carga útil do dos dados (*payload)*, e por possuir um tipo de ataque diferente dos demais *datasets* encontrados, que é o ataque de força bruta. Este *dataset* também é fácil de trabalhar pois quando aberto no Wireshark, facilita a análise e identificação dos pacotes maliciosos.

2.2.4 ISCX IDS 2012

Em (SHIRAVI, 2012), foi proposta uma nova base de dados, a UNB ISCX IDS 2012, que foi gerada a partir de tráfego real em um ambiente monitorado da Universidade de New Brinswick, no Canadá. A ideia deste *dataset* é baseada em perfis que contêm descrições detalhadas de intrusões e modelos de distribuição abstrata para aplicativos e protocolos. Vestígios do rastreamento foram analisados para criar perfis para agentes que geram tráfego real como HTTP, SMTP, SSH, IMAP, POP3 e FTP. Então, um conjunto de orientações é estabelecido para delinear *datasets* validos, que constituem a base para a geração de perfis.

Estas orientações são vitais para a eficácia do conjunto de dados em termos de realismo, capacidade de avaliação, integralidade e atividades maliciosas. Os perfis são utilizados em um experimento para gerar o *dataset* em um ambiente de campo. Diversos cenários de ataques em vários estágios foram posteriormente realizados para fornecer a parte anômala do *dataset*.

Este conjunto de dados têm as seguintes características:

Rede e tráfego realista: Um *dataset* não pode apresentar quaisquer propriedades involuntárias, tanto para tráfego normal ou malicioso. Por esta razão, é necessário que o dataset seja o mais realista possível.

Labeled dataset: Um dataset *labeled* é de extrema importância na avalição de vários mecanismos de detecção de intrusão. Criar um *dataset* em um ambiente controlado e determinístico, permite a distinção de atividade maliciosa de tráfego normal, eliminando o processo manual de separação.

Pacote de captura completo: Preocupação com a privacidade dos dados da rede tem sido um dos principais obstáculos para a segurança pois os provedores destes dados tem relutância para compartilhar as informações. A maioria desses dados são utilizados internamente, o que limita os pesquisadores da precisão para avaliar e comparar os seus sistemas, pois a remoção do *payload* resulta em uma diminuição significativa da utilização do *dataset*. Este *dataset* tem como objetivo principal gerar tráfego de rede em um ambiente controlado, preservando a naturalidade do conjunto de dados resultante.

Diversos cenários de invasão: Ataques tem aumentado com frequência, tamanho, variedade e complexidade nos últimos anos. O escopo de ameaças também mudou em esquemas mais complexos, incluindo serviço e aplicação de ataques direcionados. Estes ataques podem causar interrupções muito mais grave do que a tradicional tentativa de força bruta e também necessitam de uma profunda visibilidade de serviços IP e aplicativos para sua detecção. Com diversos cenários, o objetivo deste *dataset* é a realização de um conjunto diversificado de ataques de vários estágios, cada um dirigido para as tendências recentes em relação a ameaças.

O UNB ISCX IDS 2012 *dataset* inclui o pacote completo em formato *pcap*, que juntamente com os perfis relevantes estão disponíveis para os investigadores através de requisições por e-mail. Ele consiste de 7 arquivos com atividades de tráfego de rede (normal e malicioso). Na Tabela 5 podemos observar as características de cada arquivo:

Tabela 5: Características de cada arquivo

Dia	Data	Descrição	Tamanho (GB)
Sexta-feira	11/06/2010	Atividade normal. Nenhuma atividade maliciosa.	16.1
Sábado	12/06/2010	Atividade normal. Nenhuma atividade maliciosa.	4.22
Domingo	13/06/2010	Infiltrando a rede a partir de dentro + Atividade normal.	3.95
Segunda-feira	14/06/2010	Negação de serviço HTTP + Atividade normal.	6.85
Terça-feira	15/06/2010	Ataques distribuídos de negação de serviço usando um IRC Botnet	23.4
Quarta-feira	16/06/2010	Atividade normal. Nenhuma atividade maliciosa.	17.6
Quinta-feira	17/06/2010	Ataque de força bruta SSH + Atividade normal.	12.3

Fonte: (SHIRAVI *et al*., 2012)

2.3 Criptografia

Criptografia vem das palavras gregas que significam “escrita secreta”. Uma cifra é uma transformação de caractere por caractere ou de bit a bit, sem levar em conta a estrutura linguística da mensagem. Ela utiliza um conjunto de métodos que servem para transformar um texto claro em texto cifrado que tem a aparência de um texto gerado aleatoriamente sem sentido algum. A ação de transformar dados para uma forma ilegível é denominada cifra ou cifragem, e busca garantir a privacidade, mantendo a informação oculta de pessoas não autorizadas, mesmo que estas possam visualizar os dados cifrados. O processo inverso da cifragem é chamado de decifragem. Quando utilizamos o processo de cifragem e decifragem, necessitamos de informações confidenciais, chamadas chaves (STALLINGS, 2014). Existem dois tipos de chaves:

Chave Simétrica: É também conhecida como criptografia de chave privada. O emissor usa uma chave para cifrar a mensagem, e o receptor utiliza a mesma chave para decifrá-la.

Chave Assimétrica: É também conhecida como criptografia de chave pública. Neste tipo de criptografia, usamos duas chaves distintas, de modo a obtermos comunicação segura através de canais de comunicação inseguros. Trata-se de uma técnica de criptografia assimétrica pelo fato de usar um par de chaves distintos.

A importância da chave dentro da criptografia, se dá ao sigilo e cuidado para que ninguém sem autorização possa obtê-la de forma indevida e colocando em risco todo o

algoritmo. Um outro fator importante é a quantidade de *bits* aplicados na codificação. Começando do básico, uma chave de 8 *bits* é capaz de gerar 256 combinações diferentes. Já uma chave de 128 *bits* é cerca de 275 milhões de vezes mais difícil de ser quebrada do que uma de 90 *bits*, por exemplo. (TANENBAUM, 2003).

Podemos utilizar chaves diferentes para informações diferentes. Desta forma, mesmo que um intruso consiga descobrir uma chave, ele não conseguirá obter todas as informações. São utilizadas diversas técnicas para gerar as chaves. Entre elas, podemos destacar o uso de sistema gerador de números aleatórios. Estes sistemas funcionam, agrupando números de diferentes tipos de entrada imprevisível. Uma chave deve ser formada por números ou caracteres alfanuméricos sem coerência alguma, ou seja, ela deve ser aleatória para dificultar a quebra da mesma (STALLINGS, 2014).

2.3.1 Algoritmos criptográficos

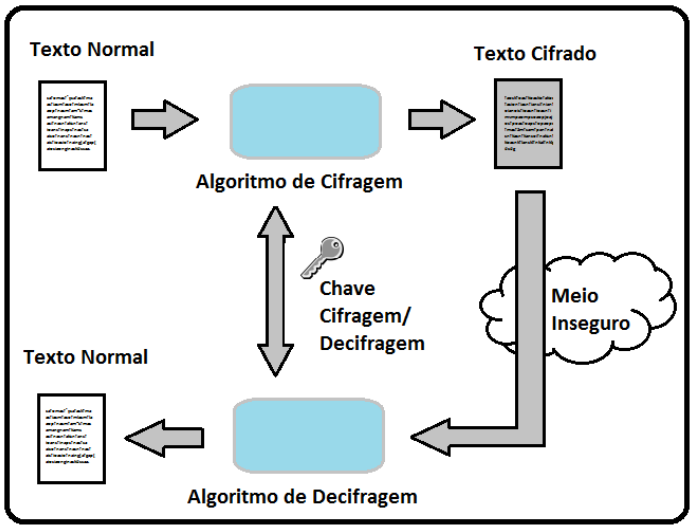
Os algoritmos criptográficos podem ser implementados em hardware (para se obter performance) ou em software (para se obter flexibilidade), embora a maior parte de nosso tratamento esteja relacionado aos algoritmos e protocolos, que são independentes da implementação real (TANENBAUM, 2003). Podemos dizer que um algoritmo de criptografia é um procedimento matemático que contém uma entrada (dados a serem cifrados), efetua um processamento matemático com base em uma chave, e gera uma saída (KUROSE *et al*., 2010).

2.3.1.1 Criptografia simétrica

A criptografia de chave simétrica utiliza a mesma chave para cifragem e decifragem. Logo, a chave deve ser conhecida tanto pelo remetente quanto pelo destinatário da mensagem. A maior dificuldade do método é a distribuição segura das chaves. Em um sistema criptográfico de chave simétrica, a chave é apenas um número qualquer que tenha um tamanho correto e que o selecione aleatoriamente (BURNETT, 2002).

Quando uma pessoa quer se comunicar de forma segura com outra pessoa, ela deve passar primeiramente a chave utilizada para cifrar a mensagem. Este processo é chamado de “distribuição de chaves”, e como a chave é o principal elemento de segurança para o algoritmo, ela deve ser transmitida por um meio seguro. Na Figura 3 é ilustrado como funciona a criptografia simétrica.

Figura 3: Esquema de criptografia simétrica



Fonte: (Adaptado de: MORENO, 2005)

Existem duas formas de se cifrar dados. Através de cifra de bloco ou cifra de fluxo.

Cifra de fluxo: Operam sobre cada bit da mensagem individualmente. Estes algoritmos são úteis quando o transmissor precisa enviar para o receptor um fluxo contínuo de dados criptografados, como por exemplo, um computador que está permanentemente transmitindo dados cifrados para outro computador. Uma operação matemática L é realizada com cada bit da mensagem separadamente de forma que se altere o valor, cifrando-a. A chave de cifragem deve ser constituída de bits que variam no tempo. Ela deve ser do mesmo tamanho da mensagem. Na decifragem o processo é semelhante ao de cifragem, bastando apenas aplicar a operação inversa L_{inv} (VELOSO, 2002).

Cifra de bloco: É definido para o algoritmo um tamanho de fragmento de dado para cifrar ou decifrar, ele divide o texto simples em blocos e opera de maneira independente (BURNETT, 2002). Suponhamos que um texto tenha 227 bytes e é utilizado uma cifragem que opera com blocos de 16 bytes. O algoritmo irá utilizar os 16 primeiros bytes, cifrar e depois os próximos 16 bytes e assim sucessivamente. Porém, ele só opera com 16 bytes. Para cifrar os últimos 3 bytes, será adicionado bytes extras no último bloco para torna-lo completo. O algoritmo de decifragem dos dados deve ser capaz de reconhecer e ignorar esse preenchimento (VELOSO, 2002).

Uma cifra de bloco usa um bloco de texto de tamanho fixo com comprimento de *b* bits e uma chave como entrada, e produz um bloco de *b* bits de texto cifrado. Se a quantidade de texto claro a ser cifrado tiver mais de *b* bits, então a cifra de bloco ainda pode ser usada quebrando o texto claro em blocos de *b* bits. Quando vários blocos de texto claro são cifrados

com a mesma chave, surgem diversas preocupações com a segurança. Para empregar uma cifra de bloco em diversas aplicações, cinco modos de operação foram definidos pelo NIST. Basicamente, um modo de operação é uma técnica para melhorar o efeito de um algoritmo criptográfico ou adaptar o algoritmo para uma aplicação, como a de uma cifra de bloco a uma sequência de blocos de dados ou fluxo de dados. Os cinco modos abrangem praticamente todas as aplicações possíveis da criptografia para as quais uma cifra de bloco poderia ser usada. Esses modos são utilizados com qualquer cifra de bloco simétrica, incluindo DES e AES (STALLINGS, 2014). Os modos são resumidos na Tabela 6.

Tabela 6: Modos de operação de cifra de bloco

Modo	Descrição	Aplicação típica
Electronic codebook (ECB)	Cada bloco de bits de texto claro é codificado independentemente usando a mesma chave.	- Transmissão segura de valores isolados (por exemplo, uma chave de encriptação)
Cipher block chaining (CBC)	A entrada do algoritmo de encriptação é o XOR dos próximos 64 bits de texto claro e os 64 bits anteriores de texto cifrado.	- Transmissão de uso geral orientada a bloco - Autenticação
Cipher feedback (CFB)	A entrada é processada *s* bits de cada vez. O texto cifrado anterior é usado como entrada para o algoritmo de encriptação a fim de produzir saída pseudoaleatória, que é aplicada a um XOR com o texto claro para criar a próxima unidade de texto cifrado.	- Transmissão de uso geral orientada a fluxo - Autenticação
Output feedback (OFB)	Semelhante ao CFB, exceto que a entrada do algoritmo de encriptação é a saída DES anterior, e são usados blocos completos.	- Transmissão orientada a fluxo por canal com ruído (por exemplo, comunicação por satélite)
Counter (CTR)	Cada bloco de texto claro é aplicado a um XOR com um contador encriptado. O contador é incrementado para cada bloco subsequente.	- Transmissão orientada a bloco de uso geral - Útil para requisitos de alta velocidade

Fonte: (STALLINGS, 2014)

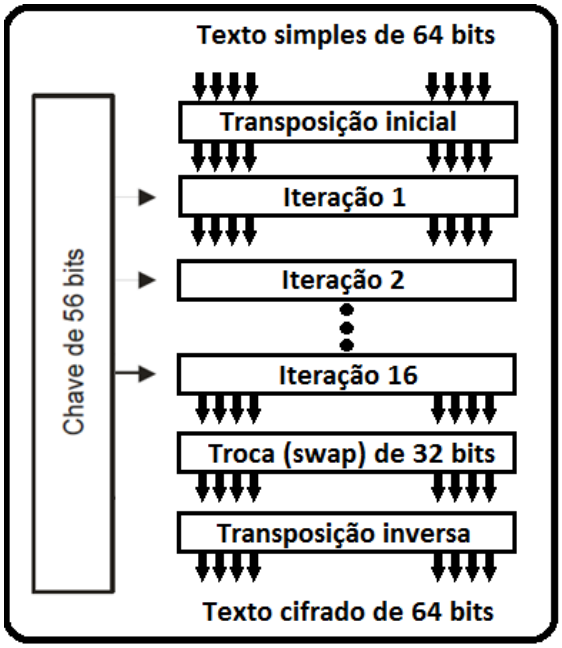
2.3.1.1.1 DES – *Data Encryption Standard*

O DES é o algoritmo simétrico mais difundido no mundo. Criado pela IBM em 1977, com um tamanho de chave de 56 bits, relativamente pequeno para os padrões atuais. Por muitos anos criptografia e DES foram sinônimas. Foi adotado rapidamente pela mídia não digital, como nas linhas telefônicas públicas. Com o passar dos anos, empresas como *International Flavors and Fragances* começaram a utilizar o DES para proteger as transmissões por telefone e outros dados importantes para ela. O setor de informática também adotou o DES para uso em produtos de segurança. Em sua forma original do DES, já não é mais tão segura, porém, em uma forma modificada ela ainda é bastante útil (STALLINGS, 2014).

O algoritmo DES é composto de operações simples, como permutações, substituições, XOR (operação lógica que produz saída verdadeira somente quando as entradas diferem) e deslocamentos. Para criptografar os dados, o DES divide a mensagem em blocos de 64 *bits* e retorna blocos de texto cifrado do mesmo tamanho. O algoritmo, parametrizado por uma chave de 56 *bits*, tem 19 estágios distintos, sendo que o primeiro deles é uma transposição independente da chave no texto simples de 64 *bits*, e no último estágio acontece o processo inverso dessa transposição. O penúltimo estágio troca os 32 *bits* mais à esquerda pelos 32 *bits* mais à direita. Os 16 estágios restantes são iterações que cada bloco de 64 *bits* sofrerá, porém cada uma delas com uma chave diferente. Antes de cada iteração, a chave é particionada em duas unidades de 28 *bits*, sendo cada uma delas girada à esquerda um número de *bits* que depende do número de iterações. Utiliza-se ainda um parâmetro **k** que é derivada da girada, pela aplicação de mais uma transposição de 56 *bits* sobre ela. Em cada rodada, um subconjunto de 48 *bits* dos 56 *bits* é extraído e permutado (TANENBAUM, 2003).

O DES foi projetado para permitir que a decodificação fosse feita com a mesma chave de codificação, uma propriedade necessária para algoritmo de chave simétrica. Porém, as etapas são simplesmente executadas na ordem inversa (TANENBAUM, 2003). A Figura 4 ilustra um esboço geral do funcionamento do algoritmo DES.

Figura 4: Esboço geral do algoritmo DES



Fonte: (Adaptado de: TANENBAUM, 2003)

Uma técnica às vezes utilizada para tornar o DES mais forte é chamada *whitening*, que consiste em operações XOR entre uma chave aleatória de 64 *bits* e cada bloco de texto simples, antes de sua entrega ao DES e depois uma operação XOR entre uma segunda chave de 64 *bits* e o texto cifrado resultante, antes de sua transmissão. O *whitening* pode ser removido com facilidade pela execução das operações inversas, para isto, o receptor deve ter as chaves de branqueamento. Esta técnica acrescenta mais *bits* ao tamanho da chave (TANENBAUM, 2003).

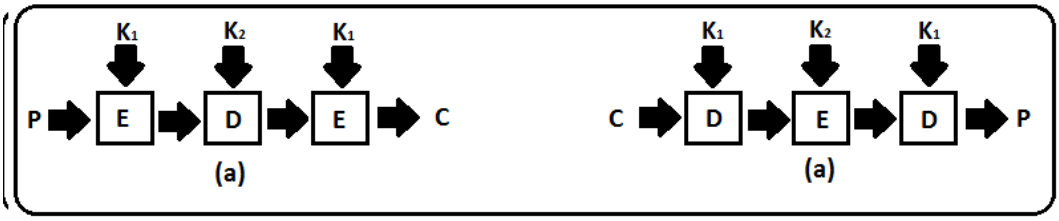
Em 1997, dois pesquisadores de criptografia de Standford, Diffie e Hellman (1977), projetaram uma máquina para decifrar o DES e estimaram que ela poderia ser montada por um custo de 20 milhões de dólares. Com base em um pequeno trecho de texto simples e no texto cifrado correspondente, essa máquina poderia descobrir a chave através de uma pesquisa exaustiva do espaço de chaves de 2^{56} entradas em menos de um dia. Atualmente, essa máquina custaria bem menos de um milhão de dólares (TANENBAUM, 2003).

Uma equipe dirigida por John Gilmore (GILMORE, 2017), construiu uma máquina que podia analisar todo o espaço de chaves de 56 *bits* de DES. Em 17 de julho de 1998, foi anunciado que havia conseguido quebrar uma chave de 56 *bits* em 48 horas. O computador que conseguiu essa proeza era chamado de *DES Key Search Machine*. A solução encontrada para este problema foi criar um algoritmo com uma chave maior. Foi desta forma que surgiu o Triple-DES (TANENBAUM, 2003).

2.3.1.1.2 Triple-DES

O algoritmo Triple-DES é apenas o DES com duas chaves de 56 *bits* aplicadas. É uma variação do DES que utiliza três cifragens em sequência, empregando chaves com tamanhos de 112 ou 168 *bits*, sendo recomendada no lugar do DES desde 1993. Dada uma mensagem em texto claro, a primeira chave é usada para cifrar a mensagem em DES e a segunda para decifrar o DES da mensagem cifrada. Como a segunda chave não é correta para decifrar, essa decifragem apenas embaralha ainda mais os dados. A mensagem duplamente embaralhada é, então, cifrada novamente com a primeira chave para se obter o texto cifrado final. Este procedimento em três etapas é chamado de Triple-DES. Utilizando duas chaves os criptográficos mais experientes concordam que 112 *bits* seriam suficientes para aplicações comerciais. O uso de 168 *bits* só criaria overhead desnecessário de gerenciar e transportar outra chave, com pouco ganho real. A Figura 5 ilustra um esboço do funcionamento do Triple-DES (TANENBAUM, 2003).

Figura 5: Criptografia tripla usando o DES



Fonte: (Adaptado de: TANENBAUM, 2003)

2.3.1.1.3 AES – *Advanced Encryption Standart*

O padrão de cifração avançado (AES) foi publicado como padrão federal de processamento de informações (FIPS 192). A intenção é substituir o DES e o triplo DES por um algoritmo mais seguro e eficiente. Devido à evolução das máquinas, uma chave com apenas 56 *bits* já não fornecia a segurança necessária. Então um órgão do departamento de comércio dos Estados Unidos chamado *National Institute of Standards and Technology* (NIST) patrocinou um concurso de criptografia, com o objetivo de conseguir um algoritmo para ser utilizado como o novo padrão. O NIST estabeleceu como requisitos:

- **Segurança forte:** o algoritmo projetado deve suportar ataques futuros;

- **Projeto simples:** facilitar a análise e certificação matemática de segurança oferecida pelo algoritmo;
- **Desempenho:** razoavelmente bom em uma variedade de plataformas, variando de *Smart Cards* até servidores;
- **Não serem patenteados:** os algoritmos devem ser de domínio público e estar disponíveis mundialmente.

O algoritmo vencedor foi o Rijndael de Joan Daemen e Vincent Rijmen, sendo que a decisão oficial foi anunciada no dia 2 de outubro de 2000. A partir desta data, o cifrador passou a se chamar de AES. É o padrão atual para cifragem recomendado pelo NIST. Trabalham com chaves de 128, 192 e 256 *bits* (TANENBAUM, 2003).

O AES utiliza a substituição, permutação e rodadas assim como o DES, porém o número de rodadas depende do tamanho da chave e do tamanho do bloco, sendo 10 para cada chave de 128 *bits* com blocos de 128 *bits*, passando para 14 no caso da maior chave ou do maior bloco. No entanto, diferente do DES, todas as operações envolvem *bytes* inteiros, para permitir implementações eficientes, tanto em *hardware* como em *software* (TANENBAUM, 2003).

Este algoritmo é classificado como um cifrador de bloco com tamanho de bloco e chave variáveis entre 128, 192 e 256 *bits*, o que significa que se pode ter tamanho de blocos com tamanhos de chaves diferentes. Em função do tamanho do bloco e chaves, determina-se a quantidade de rodadas necessárias para cifrar e decifrar (MORENO, 2005).

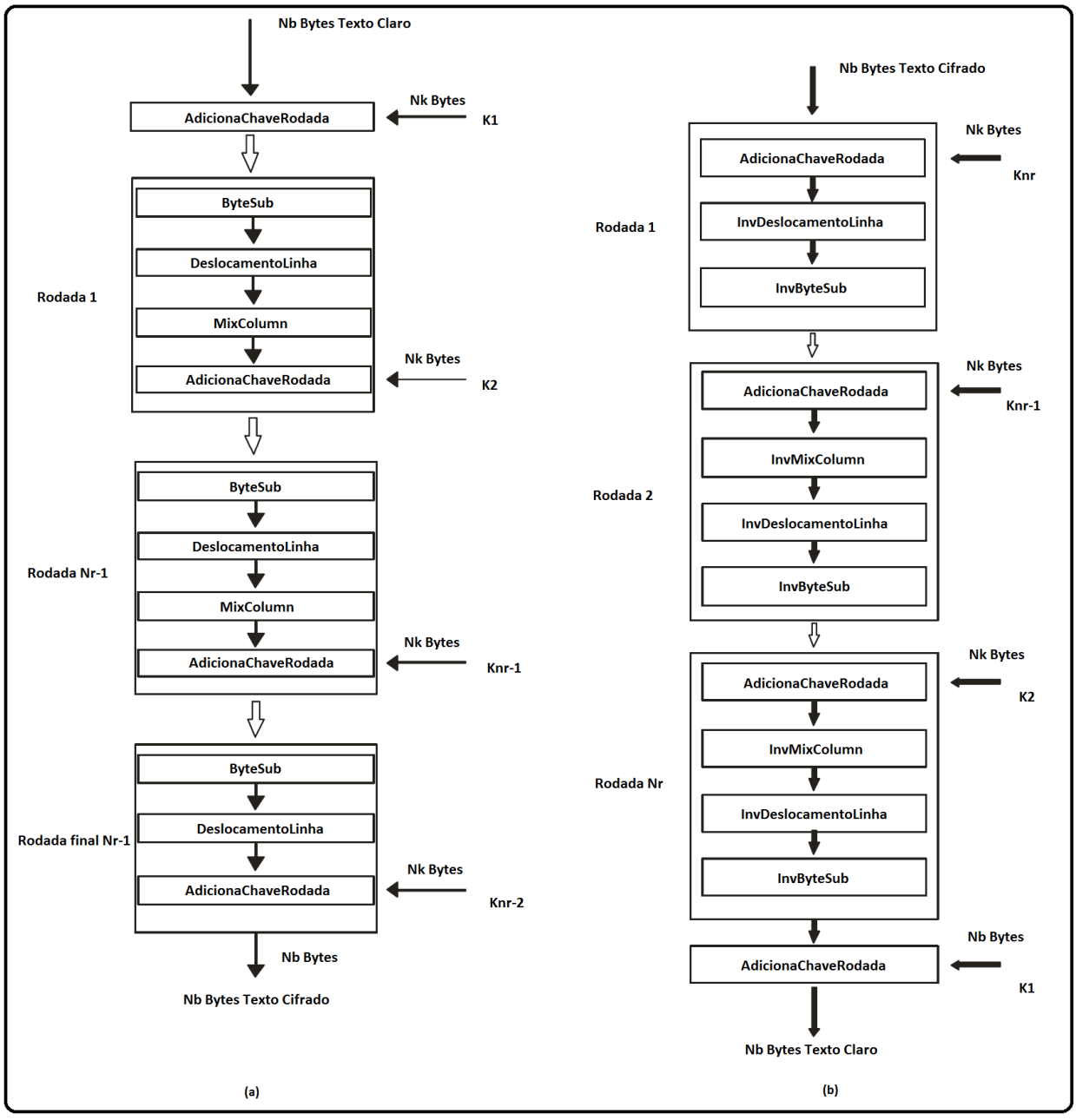
Ele opera com um determinado número de 32 *bits*, que são ordenados em colunas de 4 são ordenados em colunas de 4 *bytes* denominados **Nb**. Os valores possíveis são de 4, 6 e 8 equivalentes a blocos de 128, 192 e 256 *bits*. Logo, sempre que um **Nb** for referido, significa que se tem **Nb** x 32 *bits* de tamanho de bloco de dados. A chave é agrupada da mesma forma que o bloco de dados, em colunas, sendo representado pela sigla **Nk**. Baseado nos valores que **Nb** e **Nk** podem assumir é que se determina a quantidade de rodadas a serem executadas, identificada pela sigla **Nr**. No processo de cifragem e decifragem, as funções utilizadas não são as mesmas, como ocorre na maioria dos cifradores. Cada bloco ou estado é sujeitado durante o processo para cifrar as seguintes iterações:

- **Substituir bytes (SubByte)**: os *bytes* de cada bloco são substituídos por seus equivalentes em uma tabela de substituição (S-BOX);
- **Deslocar linhas (ShiftRow)**: nesta etapa os *bytes* são rotacionados em grupos de quatro são rotacionados em grupos de quatro *bytes;*

- **Misturar colunas (MixColumn)**: cada grupo de quatro *bytes* sujeita-se a uma multiplicação modular, o que proporciona a cada byte do grupo influenciar todos os outros *bytes*;
- **Adicionar chave de rodada (AddRoundKey)**: o bloco de dados é alterado por meio da subchave da rodada, a qual possui o mesmo tamanho do bloco, que realiza uma operação XOR com o bloco inteiro.

Por meio da análise de uma visão macro do AES, sabendo que Nb, Nk e Nr possuem valores de acordo com o tamanho de bloco e chave a serem utilizados, é possível observar que na Figura 6a (algoritmo de cifragem) a última iteração é diferente das demais. Na Figura 6b, vemos que o algoritmo poderia ser considerado como uma sequência de transformações matemáticas e o seu processo reverso consiste na aplicação da sequência inversa à original. O processo de expansão de chaves são os mesmos, porém as funções *SubByte*, *ShiftRow* e *MixColumn* necessitam ser as suas inversas matemáticas para realizar o processo de decifragem.

Figura 6: (a) Algoritmo de Cifragem do AES. (b) Algoritmo de Decifragem do AES



Fonte: (Adaptado de: MORENO, 2005)

Acima foram detalhados os algoritmos que serão utilizados neste trabalho. Podemos analisar outros algoritmos de chave privada ou criptografia simétrica de forma resumida na Tabela 7.

Tabela 7: Principais algoritmos de chave privada ou criptografia simétrica

Algoritmo	Bits	Descrição
AES	128	O Advanced Encryption Standard (AES) é uma cifra de bloco, anunciado pelo National Institute of Standards and Technology (NIST) em 2003, fruto de concurso para escolha de um novo algoritmo de chave simétrica para proteger informações do governo federal, sendo adotado como padrão pelo governo dos Estados Unidos, é um dos algoritmos mais populares, desde 2006, usado para criptografia de chave simétrica, sendo considerado como o padrão substituto do DES. O AES tem um tamanho de bloco fixo em 128 bits e uma chave com tamanho de 128, 192 ou 256 bits, ele é rápido tanto em software quanto em hardware, é relativamente fácil de executar e requer pouca memória.
DES	56	O Data Encryption Standard (DES) foi o algoritmo simétrico mais disseminado no mundo, até a padronização do AES. Foi criado pela IBM em 1977 e, apesar de permitir cerca de 72 quadrilhões de combinações, seu tamanho de chave (56 bits) é considerado pequeno, tendo sido quebrado por "força bruta" em 1997 em um desafio lançado na Internet. O NIST que lançou o desafio mencionado, recertificou o DES pela última vez em 1993, passando então a recomendar o 3DES.
3DES	112 ou 168	O 3DES é uma simples variação do DES, utilizando o em três ciframentos suscessivos, podendo empregar uma versão com duas ou com três chaves diferentes. É seguro, porém muito lento para ser um algoritmo padrão.
IDEA	128	O International Data Encryption Algorithm (IDEA) foi criado em 1991 por James Massey e Xuejia Lai e possui patente da suíça ASCOM Systec. O algoritmo é estruturado seguindo as mesmas linhas gerais do DES. Mas na maioria dos microprocessadores, uma implementação por software do IDEA é mais rápida do que uma implementação por software do DES. O IDEA é utilizado principalmente no mercado

		financeiro e no PGP, o programa para criptografia de e-mail pessoal mais disseminado no mundo.
Blowfish	32 a 448	Algoritmo desenvolvido por Bruce Schneier, que oferece a escolha, entre maior segurança ou desempenho através de chaves de tamanho variável. O autor aperfeiçoou o no Twofish.
Twofish	128	É uma das poucas cifras incluídas no OpenPGP. O Twofish é uma chave simétrica que emprega a cifra de bloco de 128 bits, utilizando chaves de tamanhos variáveis, podendo ser de 128, 192 ou 256 bits. Ele realiza 16 interações durante a criptografia, sendo um algoritmo bastante veloz. A cifra Twofish não foi patenteada estando acessível no domínio público, como resultado, o algoritmo Twofish é de uso livre para qualquer um utilizar sem restrição.
RC2	8 a 1024	Projetado por Ron Rivest (o R da empresa RSA Data Security Inc.) e utilizado no protocolo S/MIME, voltado para criptografia de e-mail corporativo. Também possui chave de tamanho variável. Rivest também é o autor dos algoritmos RC4, RC5 e RC6.
CAST	128	É um algoritmo de cifra de bloco, sendo criado em 1996 por Carlisle Adams e Stafford Tavares. O CAST-128 é um algoritmo de Feistel, com 12 a 16 iterações da etapa principal, tamanho de bloco de 64 bits e chave de tamanho variável (40 a 128 bits, com acréscimos de 8 bits). Os 16 rounds de iteração são usados quando a chave tem comprimento maior que 80 bits.

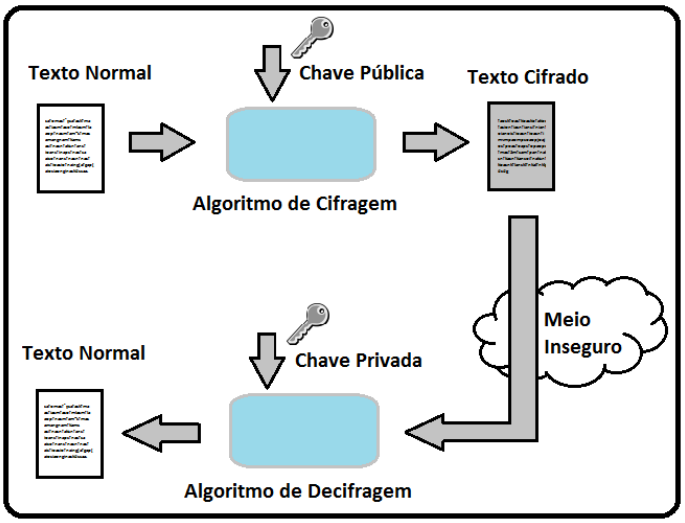
Fonte: (STALLINGS, 2014)

2.3.1.2 Criptografia assimétrica

A criptografia assimétrica, também conhecida por criptografia de chaves públicas, baseia-se no uso de pares de chaves para criptografar/descriptografar mensagens. As duas chaves são relacionadas através de um processo matemático, usando funções unidirecionais para a codificação da informação. A chave pública que, como o nome já diz, qualquer um pode conhecer e ter acesso, é usada para cifrar, enquanto a chave privada, é usada para decifrar. Uma mensagem cifrada com uma chave pública somente poderá ser decifrada com o uso da chave

privada com a qual está relacionada. A principal vantagem deste método é a segurança, pois não é necessário compartilhar a chave privada. Em contrapartida, o tempo de processamento de mensagens com criptografia assimétrica é muito maior do que com criptografia simétrica, o que pode limitar seu uso em algumas situações (BURNETT, 2002). Na Figura 7 é ilustrado como funciona o processo, com a utilização das duas chaves.

Figura 7: Esquema de criptografia assimétrica



Fonte: (Adaptado de: MORENO, 2005)

O problema deste sistema é a complexidade empregada no desenvolvimento dos algoritmos que devem ser capazes de reconhecer a dupla de chaves existentes e poder relacionar as mesmas no momento oportuno, o que acarreta num grande poder de processamento computacional (STALLINGS, 2014). Podemos analisar os principais algoritmos de chave pública ou criptografia assimétrica de forma resumida na Tabela 8.

Tabela 8: Principais algoritmos de chaves públicas ou criptografia assimétrica

Algoritmo	**Descrição**
RSA	O RSA é um algoritmo assimétrico que possui este nome devido a seus inventores: Ron Rivest, Adi Shamir e Len Adleman, que o criaram em 1977 no MIT. Atualmente, é o algoritmo de chave pública mais amplamente utilizado, além de ser uma das mais poderosas formas de criptografia de chave pública conhecidas até o momento. O RSA utiliza números primos. A premissa por trás do RSA consiste na facilidade de multiplicar dois números primos para obter um terceiro número, mas muito difícil de recuperar os dois primos a partir daquele terceiro número. Isto é conhecido como fatoração. Por exemplo, os fatores primos de 3.337 são 47 e 71. Gerar a chave pública envolve multiplicar dois primos grandes; qualquer um pode fazer isto. Derivar a chave privada a partir da chave pública envolve fatorar um grande número. Se o número for grande o suficiente e bem escolhido, então ninguém pode fazer isto em uma quantidade de tempo razoável. Assim, a segurança do RSA baseia se na dificuldade de fatoração de números grandes. Deste modo, a fatoração representa um limite superior do tempo necessário para quebrar o algoritmo. Uma chave RSA de 512 bits foi quebrada em 1999 pelo Instituto Nacional de Pesquisa da Holanda, com o apoio de cientistas de mais 6 países. Levou cerca de 7 meses e foram utilizadas 300 estações de trabalho para a quebra. No Brasil, o RSA é utilizado pela ICP-Brasil, no seu sistema de emissão de certificados digitais, e a partir do dia 1º de janeiro de 2012, as chaves utilizadas pelas autoridades certificadoras do país, passam a serem emitidas com o comprimento de 4.096bits, em vez dos 2.048bits atuais.
ElGamal	O ElGamal é outro algoritmo de chave pública utilizado para gerenciamento de chaves. Sua matemática difere da utilizada no RSA, mas também é um sistema comutativo. O algoritmo envolve a manipulação matemática de grandes quantidades numéricas. Sua segurança advém de algo denominado problema do logaritmo discreto. Assim, o ElGamal obtém sua segurança da dificuldade de calcular logaritmos discretos em um corpo finito, o que lembra bastante o problema da fatoração.

Diffie-Hellman	Também baseado no problema do logaritmo discreto, e o criptosistema de chave pública mais antigo ainda em uso. O conceito de chave pública, aliás foi introduzido pelos autores deste criptosistema em 1976. Contudo, ele não permite nem ciframento nem assinatura digital. O sistema foi projetado para permitir a dois indivíduos entrarem em um acordo ao compartilharem um segredo tal como uma chave, muito embora eles somente troquem mensagens em público.
Curvas Elípticas	Em 1985, Neal Koblitz e V. S. Miller propuseram de forma independente a utilização de curvas elípticas para sistemas criptográficos de chave pública. Eles não chegaram a inventar um novo algoritmo criptográfico com curvas elípticas sobre corpos finitos, mas implementaram algoritmos de chave pública já existentes, como o algoritmo de Diffie-Hellman, usando curvas elípticas. Assim, os sistemas criptográficos de curvas elípticas consistem em modificações de outros sistemas (o ElGamal, por exemplo), que passam a trabalhar no domínio das curvas elípticas, em vez de trabalharem no domínio dos corpos finitos. Eles possuem o potencial de proverem sistemas criptográficos de chave pública mais seguros, com chaves de menor tamanho. Muitos algoritmos de chave pública, como o Diffie-Hellman, o ElGamal e o Schnorr podem ser implementados em curvas elípticas sobre corpos finitos. Assim, fica resolvido um dos maiores problemas dos algoritmos de chave pública, o grande tamanho de suas chaves. Porém, os algoritmos de curvas elípticas atuais, embora possuam o potencial de serem rápidos, são em geral mais demorados do que o RSA.

Fonte: (STALLINGS, 2014)

2.3.2 Aplicações da criptografia

Podemos utilizar a criptografia em canais de tráfego de mensagens construídos a partir de tecnologias conhecidas. Estes sistemas podem ter diferentes níveis de complexidade. A criptografia pode ser aplicada na segurança de comunicações, identificação e autenticação. Outras aplicações envolvem sistemas para comércio eletrônico, certificação, correio eletrônico seguro, recuperação de chaves e acesso seguro a sistemas da computação (VELOSO, 2002).

Segurança de Comunicações: Aplicações envolvendo segurança de comunicações são as que mais fazem uso da criptografia. Duas pessoas podem se comunicar de forma segura

encriptando as mensagens trocadas entre elas, por exemplo, através do *WhatsApp,* que é um aplicativo de mensagens para *Smartphones*. Uma terceira pessoa que esteja interceptando estas mensagens nunca conseguirá decifrá-las (VELOSO, 2002).

Identificação e Autenticação: Identificação é o processo que identifica um indivíduo através de uma base de dados registrada. Por exemplo, quando o cliente retira dinheiro em um banco o caixa pede para que ele se identifique para verificar a identidade do proprietário da conta. Este processo também acontece de forma eletrônica com o uso da criptografia. A autenticação é semelhante a identificação, porém a autenticação é mais abrangente dado que ela não necessariamente envolve a identificação da pessoa. Ela determina se a pessoa é autorizada para aquilo em questão (VELOSO, 2002).

Comércio Eletrônico: A criptografia é algo essencial para se realizar transações eletrônicas seguras. Porém, simplesmente apresentar um número de cartão de crédito pode levar o proprietário a ser fraudado tendo o seu número de cartão de crédito usado sem sua autorização. Mecanismos de transação segura são usados na Internet, onde o número do cartão é enviado junto com outras informações de forma encriptada, permitindo assim que o pagamento seja feito de forma segura (VELOSO, 2002).

Certificação: É um esquema pelo qual agentes confiáveis (autoridades certificadoras) enviam um certificado eletrônico para um outro agente desconhecido (usuário) (VELOSO, 2002).

2.4 Tecnologias envolvidas

Este capítulo apresenta as principais tecnologias utilizadas para desenvolver este trabalho. Na Seção 2.4.1 é abordado o *software* PowerEditPcap e demais bibliotecas utilizadas. Já a Seção 2.4.2 refere-se ao software Wireshark, suas principais características e funcionalidades.

2.4.1 PowerEditPcap

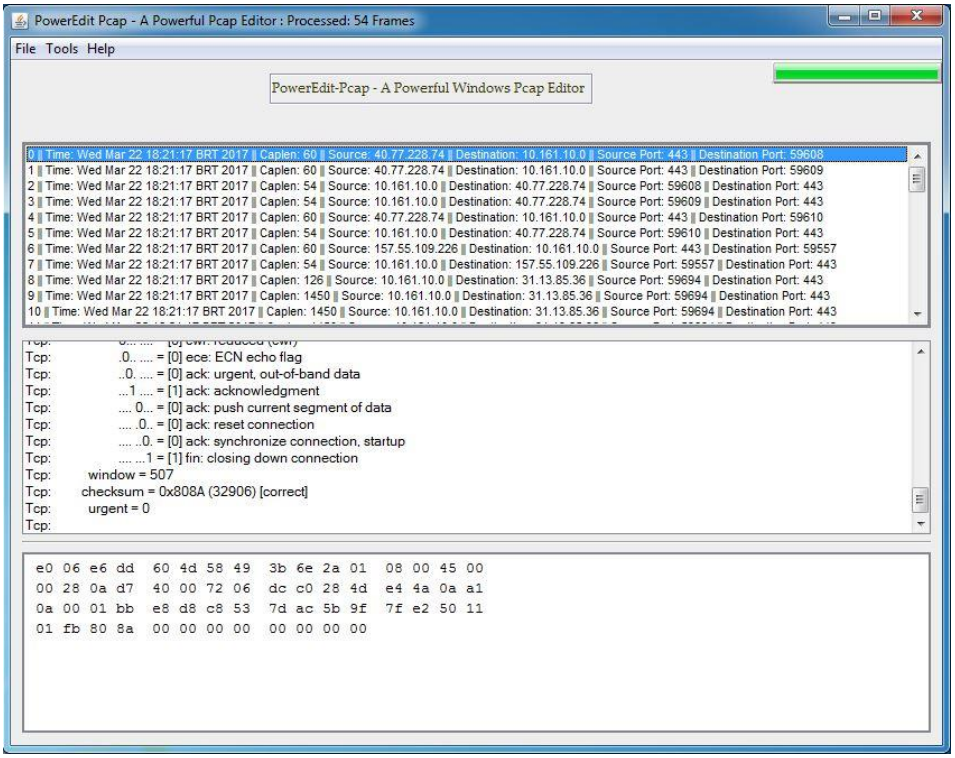
O PowerEditPcap é uma ferramenta gratuita e de código aberto. Ela permite manipular os pacotes de um arquivo *pcap*, podendo-se incluir e excluir qualquer valor dos pacotes em Hexadecimal (SOURCEFORGE, 2017). Com ela é possível:

- Editar o *header* e o *payload* dos pacotes.
- Mover *frames*

- Reorganizar os *frames*
- Reproduzir os *traces*
- Remover *traces* intermediários
- Corrigir *checksum*

Na Figura 8, a seguir, podemos ver o *software* PoweEditPcap em sua forma original.

Figura 8: Ferramenta PowerEditPcap



Fonte: (do Autor)

Este *software* utiliza uma biblioteca Java chamada jNetPcap (JNETPCAP, 2017). Esta biblioteca é de código aberto e fornece diversos recursos para se trabalhar com arquivos *pcap*, entre elas:

- Decodificar pacotes em tempo real;
- Fornece uma grande biblioteca de protocolos de rede;
- Os usuários podem facilmente adicionar suas próprias definições de protocolo usando Java SDK;

- Usa uma mistura de implementação nativa e Java para o melhor desempenho de decodificação de pacotes.

Apresentada a ferramenta PowerEditPcap, pode-se dizer que ela é fundamental para a implementação de nossa aplicação que irá criptografar o *payload* dos pacotes, pelo fato da mesma já trazer as funcionalidades básicas como abrir, editar valores dos pacotes e salvar as alterações no mesmo arquivo. A IDE escolhida para se trabalhar com o PowerEditPcap foi o NetBeans IDE (Integrated Development Environment), que é uma ferramenta de desenvolvimento modular para uma ampla gama de tecnologias de desenvolvimento de aplicações (NETBEANS IDE, 2017).

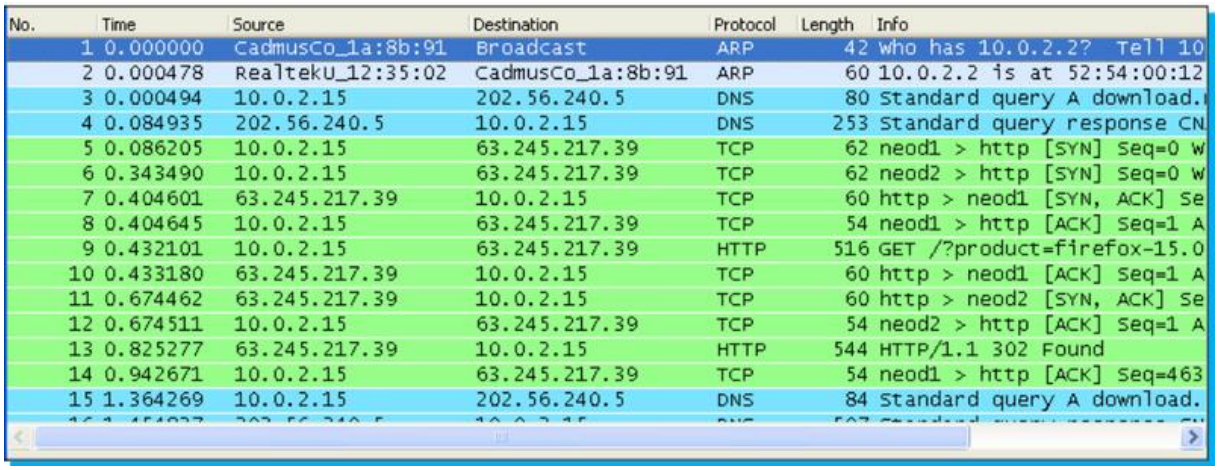
2.4.2 Wireshark

O Wireshark é uma ferramenta de análise de pacotes de rede de código aberto que captura pacotes de dados que circulam por uma rede de computadores e os apresenta de uma forma visual e compreensível. Ele suporta uma gama de protocolos que vão desde TCP, UDP, HTTP, e protocolos avançados como o Apple Talk. Possui várias opções avançadas, como a filtragem de pacotes, além da exportação de dados como resolução de nomes. É possível ainda a análise do tráfego ao vivo da rede capturando dados em tempo real, e assim gerar estatísticas sobre os protocolos, fluxo de mídia, canais de comunicação e assim por diante (SINGH, 2013).

Ao abrir o Wireshark, é possível observar que ele é dividido em 3 painéis principais: Painel de Captura de dados, Painel de Detalhes de Pacotes e por fim o Painel de Detalhes de Bytes de Pacotes. Abaixo será detalhado cada um dos painéis mencionados.

Painel de Captura de dados: mostra a captura em tempo real de pacotes de rede em uma ordem sequencial, cada linha nesta lista reflete um único pacote capturado. Este painel de visualização da informação divide-se em linhas e colunas. Cada linha representa um único pacote de dados ao passo que cada coluna representa informações adicionais sobre o pacote. A Figura 9 ilustra o Painel de Captura de dados e suas colunas.

Figura 9: Painel de Captura de dados



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	CadmusCo_1a:8b:91	Broadcast	ARP	42	Who has 10.0.2.2? Tell 10
2	0.000478	RealtekU_12:35:02	CadmusCo_1a:8b:91	ARP	60	10.0.2.2 is at 52:54:00:12
3	0.000494	10.0.2.15	202.56.240.5	DNS	80	Standard query A download.
4	0.084935	202.56.240.5	10.0.2.15	DNS	253	Standard query response CN
5	0.086205	10.0.2.15	63.245.217.39	TCP	62	neod1 > http [SYN] Seq=0 W
6	0.343490	10.0.2.15	63.245.217.39	TCP	62	neod2 > http [SYN] Seq=0 W
7	0.404601	63.245.217.39	10.0.2.15	TCP	60	http > neod1 [SYN, ACK] Se
8	0.404645	10.0.2.15	63.245.217.39	TCP	54	neod1 > http [ACK] Seq=1 A
9	0.432101	10.0.2.15	63.245.217.39	HTTP	516	GET /?product=firefox-15.0
10	0.433180	63.245.217.39	10.0.2.15	TCP	60	http > neod1 [ACK] Seq=1 A
11	0.674462	63.245.217.39	10.0.2.15	TCP	60	http > neod2 [SYN, ACK] Se
12	0.674511	10.0.2.15	63.245.217.39	TCP	54	neod2 > http [ACK] Seq=1 A
13	0.825277	63.245.217.39	10.0.2.15	HTTP	544	HTTP/1.1 302 Found
14	0.942671	10.0.2.15	63.245.217.39	TCP	54	neod1 > http [ACK] Seq=463
15	1.364269	10.0.2.15	202.56.240.5	DNS	84	Standard query A download.

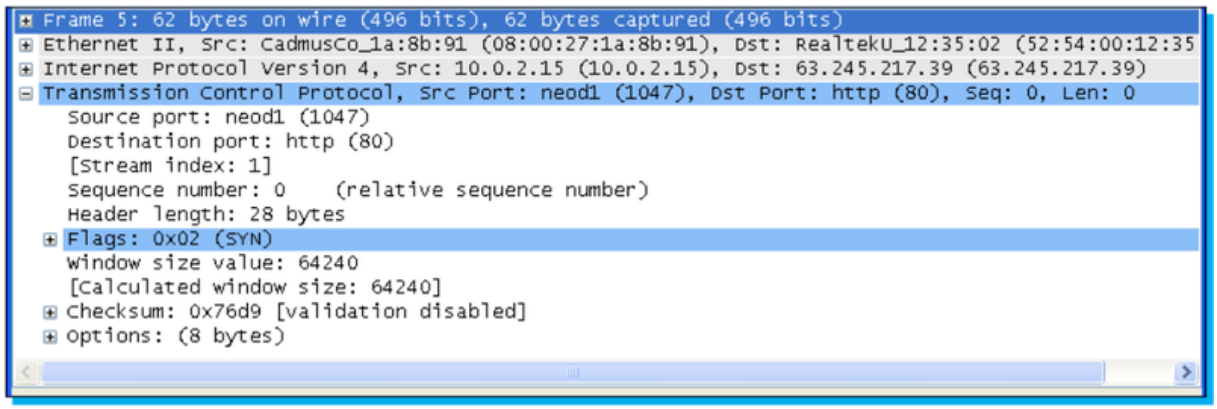
Fonte: (SINGH, 2013)

As colunas contidas no Painel de Captura de dados contêm informações específicas de cada pacote capturado, essas informações são:

- **No.:** Representa o número da sequência do pacote para identificar os pacotes com exclusividade.
- **Time:** Representa o instante de tempo que um pacote é capturado.
- **Source:** Representa o endereço IP/dispositivo de onde o pacote está vindo.
- **Destination:** Representa o endereço IP/dispositivo para onde pacote está indo.
- **Lenght:** Representa o tamanho do pacote.
- **Info:** Representa outras informações rápidas sobre o pacote.

Painel de Detalhes de Pacotes: Contêm informações detalhadas sobre os protocolos e seus diferentes parâmetros. Está organizado em uma estrutura de árvore que pode ser expandido, possibilitando uma análise detalhada das informações do pacote. Essas informações podem ser úteis para uma análise forense na rede analisada. A Figura 10 ilustra o Painel de Detalhes de Pacotes.

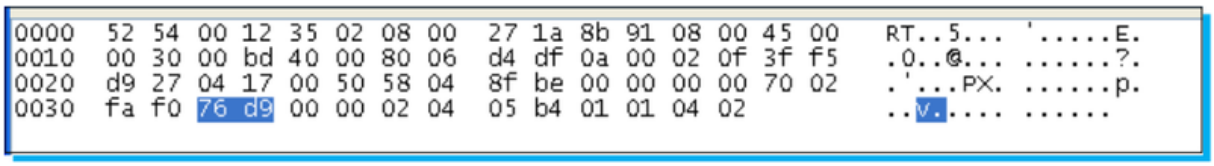
Figura 10: Painel de Detalhes de Pacotes



Fonte: (SINGH, 2013)

Painel de Detalhes de Bytes de Pacotes: Representa as informações do Painel de Detalhes do pacote em um *dump* ou em formato real do pacote. Mostra ainda as sequências de bytes do fluxo de dados. As informações são divididas em três colunas, onde a primeira coluna representa o deslocamento de dados, a segunda coluna representa os dados em valores hexadecimais, e a última representa em ASCII de informações. A Figura 11 mostra o Painel de Detalhes de Bytes.

Figura 11: Painel de Detalhes de Bytes de Pacotes



Fonte: (SINGH, 2013)

Para concluir, pode-se dizer que o Wireshark é uma ferramenta de enorme capacidade em termos de opções e análise de tráfego de rede, permitindo assim que análises detalhadas e altamente complexas como perícias forenses de rede possam ser executadas utilizando o mesmo. Esta ferramenta será de grande importância no momento em que forem feitas as validações dos *datasets*, ou seja, quando forem feitas as comparações entre os pacotes antigos (valores originais) e os pacotes alterados (com a aplicação da criptografia no *payload*).

3 ESTADO DA ARTE E TRABALHOS RELACIONADOS

Durante as pesquisas, foram encontrados alguns trabalhos que visam à detecção de ataques criptografados. A seguir, são apresentados alguns dos mais relevantes a este trabalho de conclusão. A Seção 3.1 apresenta trabalhos que justificam o desenvolvimento deste trabalho de conclusão. Já a Seção 3.2 mostra trabalhos relacionados a este trabalho.

3.1 Justificativa e objetivos

Dentre os trabalhos encontrados na literatura, o trabalho apresentado por (NEU *et al*., 2016), traz uma abordagem sobre a importância da criação de novos modelos de IDS para detecção de ataques criptografados internos no SDN - *Software Defined Networking* (FOUNDATION, 2017), pois assim como a criptografia é utilizada para segurança dos dados, ela também pode ser usada para mascarar ataques e evitar a detecção do ataque.

No trabalho proposto por Sherry *et al.*, o mesmo sugere o BlindBox, um sistema para habilitar DPI (*Deep Packet Inspection)* em cima de tráfego criptografado de dados sem a necessidade de descriptografia do mesmo. Sobre DPI, pode-se dizer que ele monitora parte do tráfego de entrada e saída dos equipamentos conectados à rede. Dessa forma, é possível realizar uma filtragem desses dados, ao constatar desvios de protocolo de rede, conteúdo que indique um ataque ou violações da política de segurança, e assim, podemos encaminhar para um destino diferente ou armazenar *logs* para análise futura (SHERRY, 2015). O BlindBox suporta aplicações como IDS, exfiltração de dados e controle dos pais. O objetivo do mesmo é utilizar DPI diretamente em tráfego criptografado através de um novo protocolo e novos esquemas de criptografia. Como o BlindBox trabalha com DPI em tráfego cifrado, seria válido testar esta ferramenta juntamente com os *datasets* cifrados neste trabalho.

Koch (KOCH *et al*, 2014) propõe uma nova arquitetura baseada no uso do conhecimento inerente de conexões de dados para cálculo de suas similaridades. Esta arquitetura não faz uso de uma fase de aprendizado, configuração complexa ou mesmo conhecimento do serviço a ser protegido. Seus resultados mostram uma eficácia de 74% quando 1% de tráfego malicioso é adicionado e 72% quando 2.7% dos pacotes são maliciosos. A taxa de alarme falso está acima de 26%. Os *datasets* propostos neste trabalho de conclusão contendo o *payload* cifrado, podem ser utilizados com essa arquitetura na tentativa de detectar ataques com tráfego criptografado.

Foroushani *et al.* apresenta em seu trabalho, um IDS com base na análise do tamanho dos pacotes e do tempo na rede. Com estas técnicas é possível analisar dados na camada de

aplicação, mas são necessárias configurações mais completas ou configurações mais detalhadas de perfis de servidores. Além disso, este trabalho apresenta altas taxas de falso positivo, que chegam a mais de 47%.

He *et al.* apresenta mais um método para melhorar a segurança na nuvem. Este método tem como objetivo detectar exfiltração de dados criptografados. Inicialmente é utilizado DPI e estimativa de entropia de amostra para identificar tráfego criptografado. Após, foi gerado o perfil de comportamento da rede para determinar o estado da encriptação. Eles selecionaram a hora da ocorrência, porta e IP de destino, protocolos da camada de rede e de aplicação do tráfego encriptado para representar comportamentos da rede. Nos experimentos, este método levou cerca de 45 segundos para identificar tráfego criptografado e as taxas de detecção ficaram acima de 90%. A precisão da determinação do estado da encriptação foi de mais de 80% e a taxa de falsos positivos foi baixa.

Alguns artigos recentes descrevem propostas para mecanismos de detecção de intrusão baseados em SDN. Jankowski *et al*. comenta sobre quatro modelos de solução:

- Método 1: baseado na análise do primeiro pacote transmitido ao Controlador e tem a intenção de detectar ataques *probe* e DoS. Utiliza SDN para detecção de tráfego anômalo.
- Método 2: este método consiste em gerenciamento de segurança para SDN baseado em lógica *fuzzy*. A execução do mesmo se dá através da avaliação do nível das *threads* e serve para detectar ataques DoS.
- Método 3: propõe a criação de perfis utilizando *sFlow* (HUANG *et al*., 2016) e *OpenFlow* (OPENFLOW, 2016). Combina estes para detecção de anomalias e mecanismos de mitigação efetivos e escaláveis em ambientes SDN afim de detectar DDoS e propagação de ataques *probe*.
- Método 4: este é um modelo leve de detecção de ataques *flooding* de DDoS utilizando NOX/*OpenFlow* baseado em coleta de estatísticas de *flow*. Este método tende a identificar ataques DDoS praticados por *botnet.*

Os métodos descritos por Jankowski podem ser utilizados para detecção de atividades maliciosas comuns, como DoS, *port scan* (varredura de porta) e tentativas de propagação de software malicioso. Porém, nenhum dos métodos utilizando soluções SDN foi capaz de detectar atividades maliciosas criptografadas.

Através deste trabalho de conclusão, busca-se criar *datasets* com pacotes cifrados, aplicando criptografia no *payload* de *datasets* existentes contendo ataques. Para isso,

desenvolveu-se uma ferramenta capaz de abrir *datasets*, extrair seu *payload* e aplicar cifragem sobre o mesmo, para que no final, seja gerado um *dataset* com pacotes maliciosos criptografados. Estas bases são importantes para que novos métodos de IDSs possam identificar ataques com fluxo cifrado.

3.2 Trabalhos Relacionados

Através de pesquisas, foram encontrados trabalhos na literatura relacionados a detecção de ataques criptografados. Na sequência, são apresentados os mais relevantes para este trabalho.

As medidas atualmente aplicadas que identificam a criptografia requerem conhecimento específico de dados indicando criptografia. Tal conhecimento pode ser, por exemplo, padrões de bytes característicos, as chamadas assinaturas. Além disso, o teste de entropia de largo alcance examina sequências de bytes na sua distribuição relativa de caracteres ocorrentes. Lyda *et al.* descreve uma abordagem para identificar *malware* criptografado com a ajuda da análise de entropia. Embora a detecção de ambos seja possível, ela não consegue distinguir entre criptografia e compressão. Jozwiak *et al.* concluiu que, além da aplicação do teste de entropia, era necessária uma verificação adicional por análise de assinatura de arquivo para poder distinguir entre criptografia e compressão. Zhang *et al.* descreve um método para detectar *botnets* adicionando análise de entropia para uma ferramenta existente de inspeção profunda de pacotes (DPI). No entanto, uma vez que outros formatos de arquivos também satisfazem uma entropia elevada, é necessário efetuar mais análises para reduzir falsos positivos.

A detecção de dados criptografados persistentemente armazenados desempenha um papel cada vez mais importante na forense digital. Isso fica claro durante a análise de sistemas de TI, quando as estruturas de dados criptografados são temporariamente descriptografados na memória principal e, portanto, podem ser acessados como um texto simples. Um método comum utilizado para detectar a presença de dados criptografados em um dispositivo de armazenamento é o cálculo da entropia. Porém, este método tem uma desvantagem significativa, pois os dados aleatórios e comprimidos têm uma entropia muito semelhante em comparação com os dados criptografados, o que gera uma taxa de falsos positivos elevada, e por esse motivo, a entropia não é muito adequada para diferenciar esses tipos de dados.

Thurner *et al*., sugere um *workflow* (fluxo de trabalho) para a detecção de estruturas de dados criptografados em um dispositivo de armazenamento e um algoritmo de classificação melhorado. A parte de classificação do *workflow* é baseada em testes estatísticos. Para

conveniência do investigador um objetivo importante é minimizar o número de estruturas de dados falsamente classificadas não criptografadas (por exemplo, os dados comprimidos são classificados como dados codificados). A abordagem para alcançar esse objetivo é combinar diferentes testes estatísticos. Uma ferramenta foi fornecida e avaliada para análise automatizada de dispositivos de armazenamento que implementa uma infinidade de testes estatísticos para a detecção melhorada de dados criptografados, em comparação com a aplicação de apenas um desses testes e o cálculo da entropia. Esta ferramenta foi capaz de distinguir de forma confiável os formatos de arquivos de alta entropia (arquivos de texto e imagem) de arquivos criptografados.

Sperotto *et al.* propõe um *dataset labeled* para detecção de intrusão *flow-based* (baseada em fluxo). A detecção de intrusão baseada em fluxo tornou-se recentemente um mecanismo de segurança promissor em redes de alta velocidade. Em relação ao *dataset*, foi proposto um conjunto de dados enriquecidos para ajuste, treinamento e avalição de IDS. No contexto específico desta experiência, o hospedeiro era um *honeypot*, que consiste em uma ferramenta que tem a função de propositalmente simular falhas de segurança em um sistema e colher informações sobre o invasor. As informações coletadas permitiram criar uma base de dados de fluxos e eventos de segurança (alertas). O trabalho também descreve um processo de correlação semi-automatizado que combina fluxos com eventos de segurança, e além disso, reflete as relações da causalidade entre os próprios eventos de segurança. Os resultados do processo de correlação mostram que pôde-se rotular mais de 98% dos fluxos no *trace* (rastreamento). A abordagem proposta permite capturar eventos de segurança inesperados, como o comportamento de um *host* comprometido. No entanto, o *trace* apresentado consiste principalmente de tráfego malicioso. Para a avaliação de um IDS, isso significa que o *dataset* permite detectar falsos negativos, mas não falsos positivos.

Na Tabela 9 é possível visualizar uma comparação entre os trabalhos relacionados e este trabalho de conclusão. Pode-se observar que o objetivo principal de estudo nos trabalhos mencionados está relacionado a novos métodos de detecção de intrusão em tráfego criptografado.

Tabela 9: Comparativo entre trabalhos estudados e o trabalho proposto

Autor	Método Detecção de Intrusão	Dataset	Classificação de Fluxo Cifrado	Detecção de Ataque Cifrado
Lyda	√	X	√	√
Jozwiak	√	X	√	√
Zhang	√	X	√	X
Thurner	√	X	√	X
Sperotto	X	√	X	X
Igor (Este trabalho)	X	√	X	√

Fonte: (do Autor)

Ao analisar a Tabela 9, observa-se que alguns autores abordam métodos de detecção de intrusão de ataques criptografados e utilizam ou proveem *datasets* para isso, outros sugerem métodos de classificação de pacotes de dados cifrados utilizando cálculo de entropia, visto que está é a primeira etapa para detectar um ataque criptografado, e outros disponibilizam *datasets* com ataques criptografados. Porém, o grande diferencial dos *datasets* gerados neste trabalho de conclusão está no fato do mesmo fornecer ataques com *payload* criptografados.

Estes *datasets* serão cifrados através de uma ferramenta desenvolvida, aplicando algoritmos de cifragem AES no conteúdo de dados (*payload)* dos pacotes maliciosos. Mais detalhes sobre a ferramenta e os métodos implementados serão apresentados no Capítulo 4.

4 APLICAÇÃO DA CRIPTOGRAFIA SOBRE OS *DATASETS*

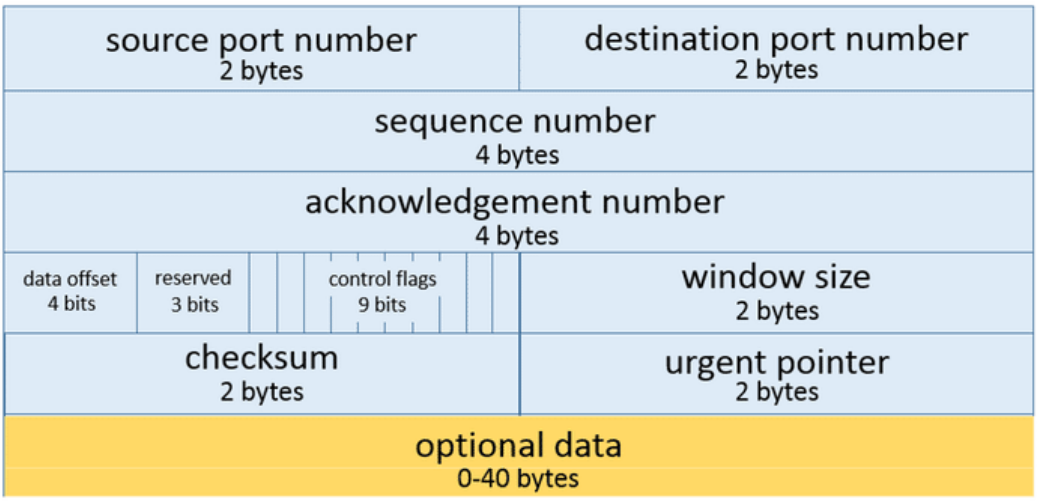
O objetivo deste trabalho é criar *datasets* contendo ataques criptografados, aplicando algoritmos de cifragem no *payload* de alguns pacotes maliciosos. Neste projeto não foi utilizado o KDDCUP99, pois o mesmo não contém os dados transmitidos. Portanto, utilizou-se os *datasets* ISCX IDS 2012 e hydra_telnet.

Disponibilizado pela Universidade de New Brinswick (UNB, 2016), do Canadá, o *dataset* ISCX IDS 2012 foi adquirido através de uma requisição feita por e-mail, pois não foi possível obtê-lo de outra forma. Esta base possui tráfego realista e diversos cenários de invasão, divididos em 7 arquivos que indicam os dias em que foram capturados. Já o *dataset* hydra_telnet utiliza ataques de força bruta, que consiste em consecutivas tentativas de verificação de senhas sobre um serviço em um determinado alvo, a qual se utiliza um banco de palavras para verificação dos dados. Estes ataques foram gerados através de uma ferramenta chamada Hydra. O protocolo utilizado para injetar os ataques foi o protocolo Telnet.

Para cifrar os pacotes, foi utilizado a criptografia simétrica, através do algoritmo AES. Optou-se por este algoritmo pois ele proporciona mais segurança e eficiência que seus antecessores. É um dos mais populares desde 2006, sendo considerado como o padrão substituto do 3DES. O AES utiliza substituição, permutação e rodadas assim como o DES, porém o número de rodadas depende do tamanho da chave e do tamanho do bloco.

Para melhor entendimento deste trabalho, faz-se necessário explicar brevemente como funciona a composição de um pacote TCP/IP. Ele é composto basicamente de duas partes: cabeçalho (*header*) e uma área de dados (*payload*). É no cabeçalho que estão as informações de tráfego e controle do protocolo. O *payload* é responsável por armazenar os dados que irão trafegar entre os *hosts* (FILHO, 2013). A Figura 12 exemplifica a composição de um cabeçalho TCP.

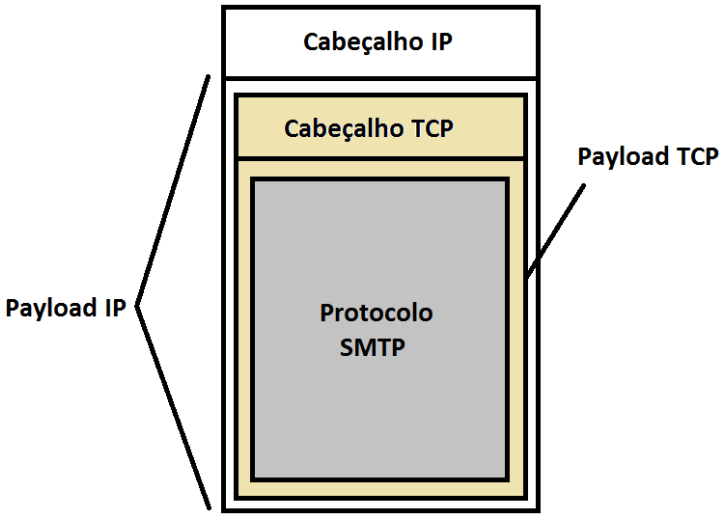
Figura 12: Composição de um cabeçalho TCP



Fonte: (KUROSE, 2010)

A primeira informação importante que podemos obter da Figura 12 é que o cabeçalho TCP, da mesma foram como ocorre com o IP, possui 32 bits de largura (de 0 a 31), o que corresponde a 4 *bytes* de largura. O protocolo IP sempre receberá outro protocolo dentro de seu *payload*. São os chamados protocolos IP. Cada protocolo tem uma importância e executa alguma função. Contudo, dois desses protocolos possuem uma função especial: transportar outros protocolos. São eles TCP e o UDP. Em outras palavras, o TCP e o UDP recebem outros protocolos nos seus *payloads*, como o SMTP. A Figura 13 mostra um exemplo de associação do IP com o TCP e o SMTP (FILHO, 2013).

Figura 13: Encapsulamento do protocolo SMTP pelo TCP



Fonte: (Adaptado de: FILHO, 2013)

Na Figura 13, ficou caracterizado o encapsulamento de protocolos. Encapsular um protocolo é colocá-lo dentro do *payload* de outro protocolo. A maioria dos outros protocolos IP, diferentemente do TCP e do UDP, não recebe outros protocolos nos seus *payloads.* Esses outros protocolos, no máximo, carregam dados comuns nos seus *payloads.*

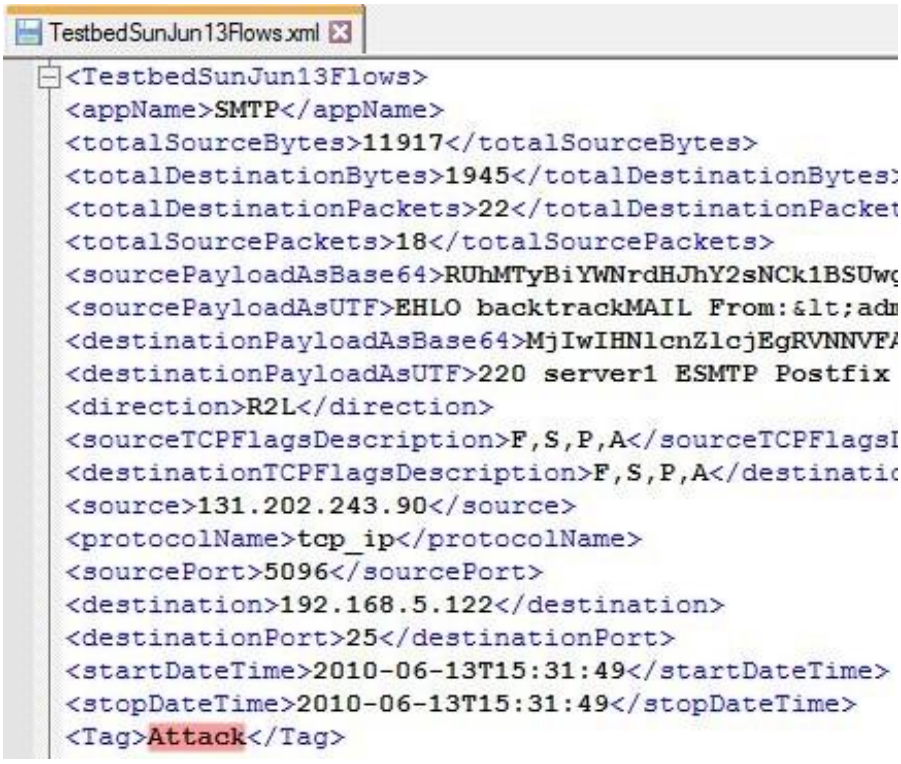
A ferramenta implementada neste trabalho deve ser capaz de criptografar somente o *payload,* ou seja, a área de dados do pacote. O desenvolvimento deste trabalho foi dividido em duas etapas e está organizada da seguinte forma: Seção 4.1 é apresentada a análise dos pacotes maliciosos, e por fim, na Seção 4.2 são apresentados os métodos implementados para a criação do *dataset* cifrado.

4.1 Análise dos pacotes maliciosos

Nesta seção, apresenta-se a análise dos pacotes maliciosos dos *datasets* envolvidos. Para isso, utilizou-se o Wireshark, que é uma ferramenta de captura e análise de pacotes de dados que circulam por uma rede e os apresenta de uma forma visual e compreensível. Também foram utilizados arquivos disponibilizados juntamente com os *datasets* para identificação dos ataques.

Para identificar os pacotes maliciosos do *dataset* ISCX IDS 2012, utilizou-se os arquivos “label_flows_xml” fornecidos juntamente com os *pcaps*, que contém informações de fluxo detalhadas em formato XML para cada um dos dias 7 dias coletados. A coluna “Tag” indica se o fluxo é normal ou é parte de um cenário de ataque. A Figura 14 apresenta um dos fluxos extraídos do arquivo XML contendo ataques.

Figura 14: Arquivo XML com fluxo de ataques



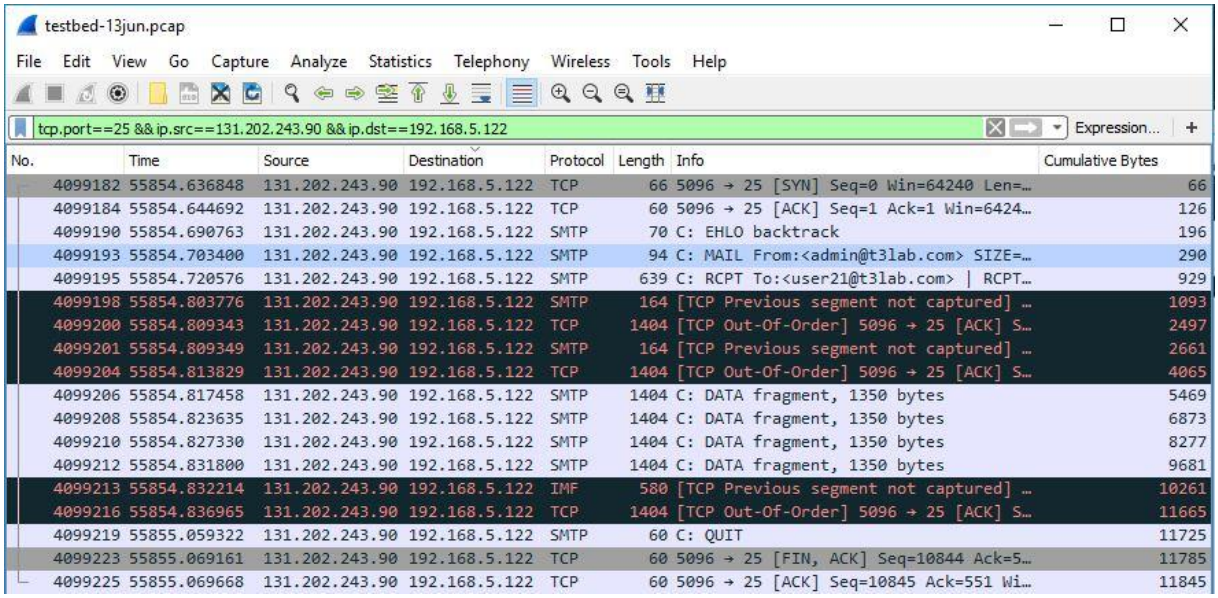
Fonte: (do Autor)

Na Figura 14 podemos observar que o ataque foi injetado através do protocolo SMTP, que por sua vez utiliza a porta 25 como destino. A "Tag" com a informação *"Attack"* identifica que neste fluxo ou período ocorreu um ataque. Como mencionado anteriormente, o *dataset* é dividido em 7 partes (dias), e para este trabalho foi selecionado o arquivo **testbed-13jun.pcap**, pelo fato do mesmo obter ataques internos e tráfego normal.

A segunda fase da análise é feita através da ferramenta Wireshark, onde é possível detalhar cada um dos pacotes do *dataset*. Após a abertura do arquivo *pcap*, aplicou-se alguns filtros para que fosse possível encontrar os 18 pacotes do fluxo de ataque. A Figura 15 mostra o resultado obtido aplicando o filtro.

tcp.port==25 && ip.src==131.202.243.90 && ip.dst==192.168.5.122

Figura 15: Conjunto de pacotes com fluxo de ataques

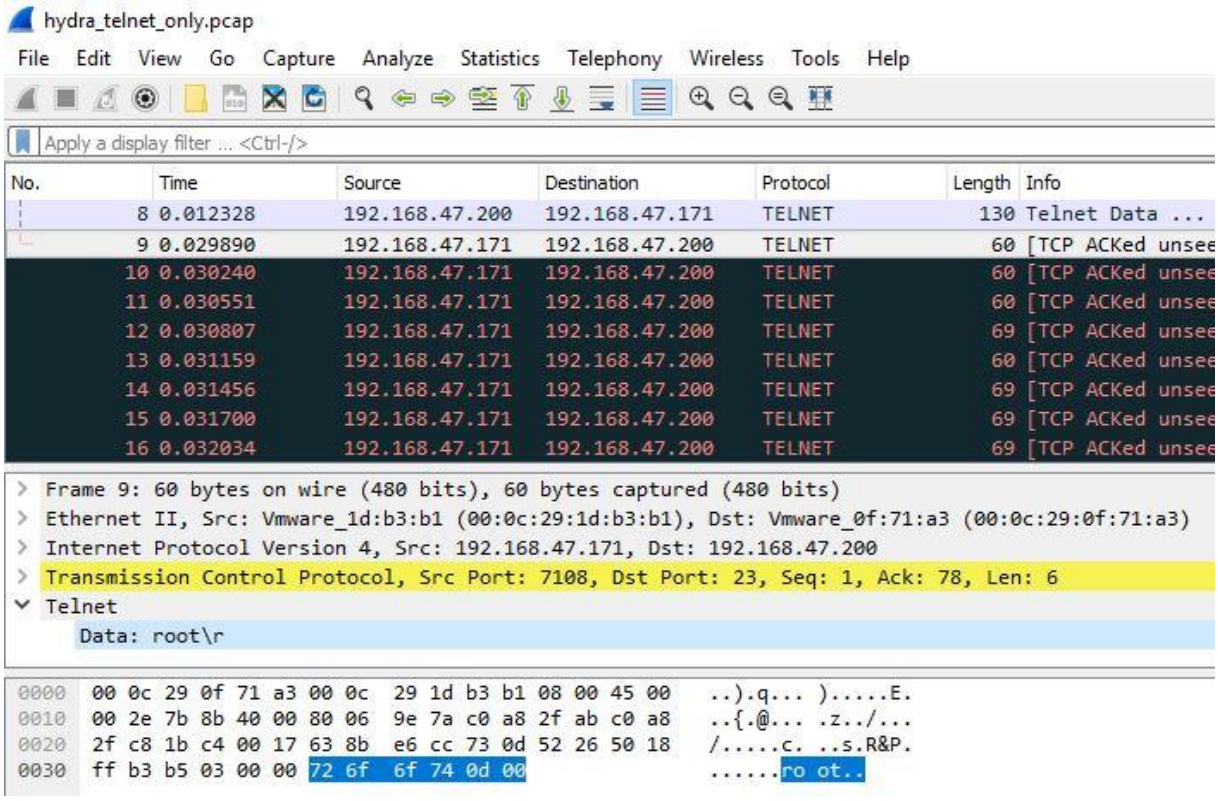


Fonte: (do Autor)

Pode-se observar que aplicando o filtro acima, obtêm-se exatamente os 18 pacotes listados no arquivo XML como sendo ataques SMTP. Após identificados, exporta-se estes 18 pacotes para um novo *pcap* chamado **testbed13junSMTPattack.pcap**, pois fica inviável trabalhar com o *dataset* completo, sendo que este contém 5763149 pacotes e dificulta a visualização e identificação dos fluxos de ataques. Com isso, temos um *dataset* compacto, e este será utilizado nos testes posteriores.

Criou-se também um novo *pcap* através do *dataset* hydra_telnet, exportando todos os pacotes do protocolo Telnet para um novo arquivo chamado **hydra_telnet_only.pcap**, este contendo ataques de força bruta conforme ilustrado na Figura 16.

Figura 16: Conjunto de pacotes com ataques de força bruta



Fonte: (do Autor)

Os dois *datasets* criados, **testbed13junSMTPattack e hydra_telnet_only,** serão utilizados para testes e validações no decorrer desde trabalho. Na próxima Seção detalharemos a implementação da ferramenta que irá criptografar o *payload* de alguns pacotes destes *datasets.*

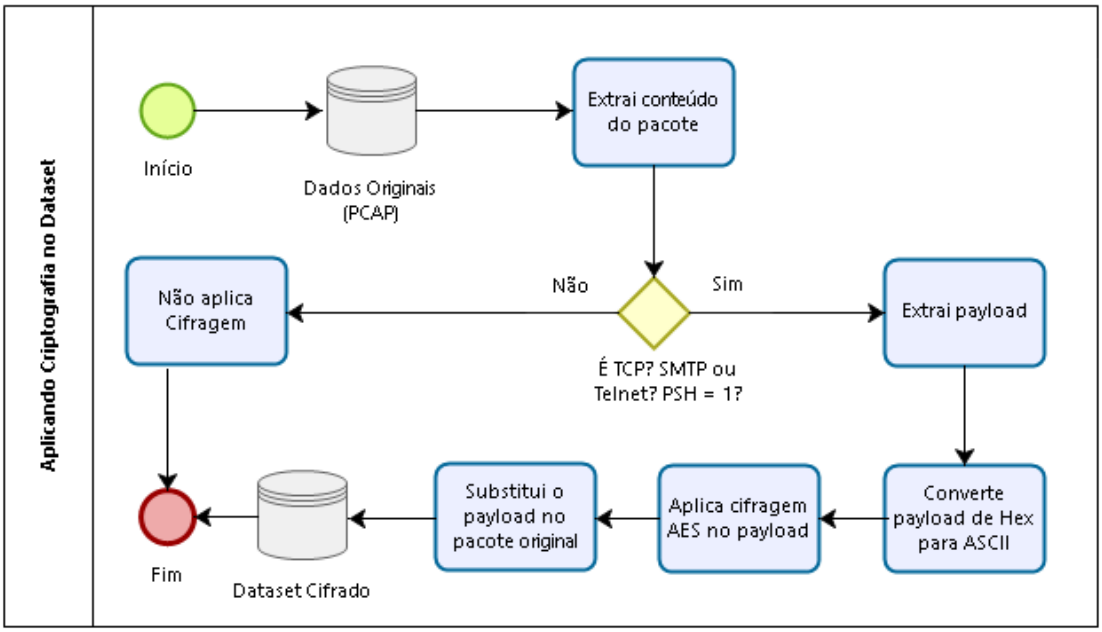
4.2 Métodos implementados para criação de *Datasets* com *payload* cifrado

Como mencionado na Seção 2.4.1, para o desenvolvimento da ferramenta de cifragem foi utilizado como base o *software* PowerEditPcap, que já traz algumas funcionalidades essenciais para este projeto, como abertura de um arquivo *pcap*, edição de valores dos pacotes e por fim, salvar as alterações feitas. Deseja-se criar uma ferramenta que cifre o *payload* de alguns pacotes dos *datasets* gerados, e para isso, foram desenvolvidos 6 métodos adicionais sobre o programa original. Algumas modificações foram feitas na tela e serão mostradas e explicadas na Seção de Testes e Resultados.

Em seguida, serão explicados cada um dos métodos desenvolvidos para cifrar o *payload* dos pacotes. O programa original já traz em uma *String* chamada **hexdump** todos os valores do pacote em hexadecimal, ou seja, ao abrir um arquivo *pcap*, o programa faz uma varredura em

todos os pacotes do arquivo e através desta variável é possível adquirir o conteúdo do pacote em hexadecimal e trabalhar com ele. Como nem todos os pacotes tem *payload*, faz-se necessário percorrer os valores do pacote a fim de identificar quando a criptografia deve ser aplicada. O fluxograma da Figura 17 ilustra todos os métodos implementados para a geração do *dataset* cifrado.

Figura 17: Fluxograma do trabalho desenvolvido

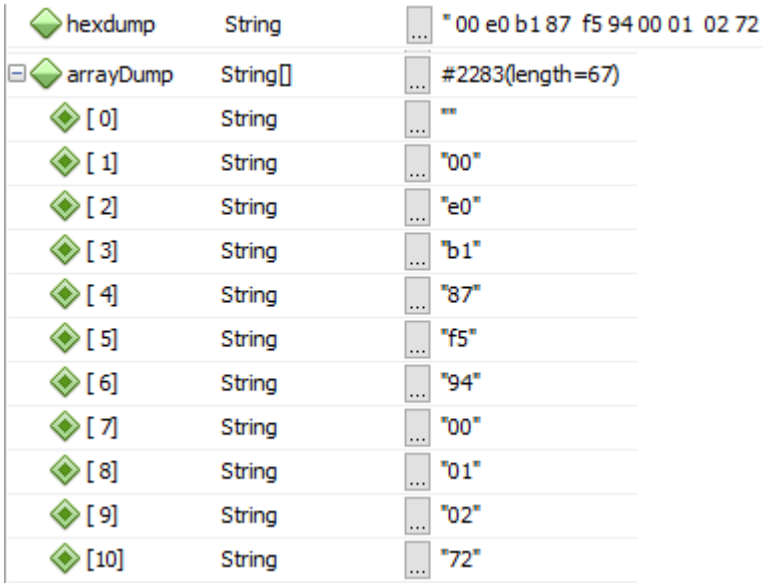


Fonte: (do Autor)

- **Método 1: Extrai conteúdo do pacote:**

Este método extrai todo o conteúdo do pacote em hexadecimal através da *String* **hexdump.** Em seguida converte-se esta *String* em um *Array* chamado **arrayDump**, para que posteriormente seja possível percorrer pelo *Array* e identificar alguns valores em determinadas posições do seu índice. A Figura 18 ilustra as variáveis **hexdump** e **arrayDump** citadas acima.

Figura 18: Conversão de String para Array



Fonte: (do Autor)

- **Método 2: Verifica se o pacote é TCP, protocolo SMTP ou Telnet e flag PSH=1:**

É através deste método que definimos se o pacote será cifrado ou se o programa irá descarta-lo, ou seja, se deve ignorá-lo e passar para o próximo pacote. Implementou-se um método de verificação baseado nos dois *datasets* gerados anteriormente, que utilizam os protocolos SMTP e Telnet para inserir seus ataques. Este método verifica se o pacote em análise é TCP, e isso é feito através de uma checagem na posição [24] do *Array*, se este conter o valor "06" indica que ele é um pacote TCP. Além disto, o método também verifica a posição [38] a fim de encontrar o valor "19", indicando que este pacote utilizou o protocolo SMTP como destino, baseado no *dataset* **testbed13junSMTPattack** que utiliza este protocolo para injetar os ataques. Se caso não identificar o pacote como SMTP, faz uma segunda checagem procurando o valor "17" também na posição [38] com o objetivo de identificar o protocolo Telnet como destino, baseado no *dataset* **hydra_telnet_only** que utiliza este protocolo para ataques de força bruta**.** E por fim, o método também verifica se o pacote contém a *flag* PSH = 1, sinalizando que existem dados no *payload* do segmento TCP em questão (FILHO, 2013), e para isso procura-se o valor "18" na posição [48] do *Array*. A Figura 19 mostra o segundo método implementado para identificar se o pacote deve ser cifrado.

Figura 19: Método para identificar pacote TCP

```
public boolean IsTcp(String[] arraydump){
    // faz a verificação se o pacote é TCP
    boolean result = false;
      if(arraydump[24].equals("06")){ //TCP
        if(arraydump[38].equals("19")){ // SMTP=19
            if(arraydump.length > 47 && arraydump[48].equals("18")){ //PSH
              result = true;
          }
        }
        else if (arraydump[38].equals("17")){ // Telnet=17
            if(arraydump.length > 54){
             result = true;
        }
      }
    }
    return result;
}
```

Fonte: (do Autor)

Utilizando como exemplo o terceiro pacote do *dataset* **testbed13junSMTPattack** (Figura 20)**,** pode-se observar que o mesmo traz todas as características que o método exige para retornar o valor **true** e indicar para o programa que o *payload* deste pacote deve ser cifrado.

Figura 20: Pacote TCP, protocolo SMTP e flag PSH

```
0000  00 e0 b1 87 f5 94 00 01  02 72 ab 55 08 00 45 00   ........ .r.U..E.
0010  00 38 7a 55 40 00 79 06  4a 23 83 ca f3 5a c0 a8   .8zU@.y. J#...Z..
0020  05 7a 13 e8 00 19 17 11  f1 6e df 40 a9 5b 50 18   .z...... .n.@.[P.
0030  3e b3 2c fc 00 00 45 48  4c 4f 20 62 61 63 6b 74   >.,...EH LO backt
0040  72 61 63 6b 0d 0a                                 rack..
```

Fonte: (do Autor)

- **Método 3: Separa cabeçalho do *payload*:**

Para que o pacote fique integro após a aplicação da criptografia, é importante que o cabeçalho fique exatamente como no pacote original, afinal, o objetivo deste trabalho é cifrar somente a área de dados. Para isso, criou-se o método **IsolateDataAndHeader** que separa o cabeçalho em uma *String* chamada **header,** e os dados transmitidos em outra *String* definida como **payload.**

Através de uma análise, foi possível observar que tanto no protocolo SMTP quanto no Telnet, o *payload* sempre começa a partir da posição [55] do *Array*, e tudo que vem antes disso

é cabeçalho. O endereço IP, que possui 4 *bytes* de largura, é o maior dado que compõe o cabeçalho de um pacote IP. Esse é o motivo pelo qual o cabeçalho IP tem 32 *bits* de largura. Logo, da posição [1] até [54] do *Array* temos informações de controle e tudo que vier depois disso trata-se dos dados transmitidos (*payload).* Portanto, quebrou-se o pacote de *Array* em duas *Strings:* **header[1]-[54]** e **payload[55]-[N]**, para que posteriormente se aplique a cifragem somente nos dados do pacote.

- **Método 4: Converte *payload* de Hexadecimal para ASCII:**

O método **convertHexToString** converte a *String* **payload** que está em valor hexadecimal para o código *American Standard Code for Information Interchange* – ASCII. Esta conversão é necessária pois o próximo método que cifra o *payload* deve receber uma *String* em texto puro.

- **Método 5: Aplica criptografia AES no *payload:***

Após ser transformado em texto claro, o *payload* é enviado para o método de criptografia**.** O algoritmo utilizado para cifragem foi o AES com o modo de operação **CTR** (*Counter*). O *Counter Mode* é um modo de funcionamento da cifra AES em que podemos computar o bloco AES a priori sem a presença dos dados a serem cifrados. Ele aplica uma sequência de blocos de entrada na cifragem, chamados de contadores, para produzir blocos cifrados que então são utilizados para realizar a operação XOR com o texto plano, ou com o texto cifrado, no caso da decifragem. A única condição imposta por este modo é que cada bloco na sequência, para uma chave igual, seja único. Por não utilizar a mensagem em si para realizar a cifragem, o modo CTR pertence ao conjunto de modos de fluxo de chave, não necessitando de *padding* ao final da mensagem cifrada. Tem-se como objetivo principal, obter o texto cifrado exatamente com o mesmo tamanho do texto plano, ou seja, uma criptografia N-N, logo o CTR foi o modo de operação compatível com este propósito. A chave utilizada para criptografar os dados foi **"0123456789abcdef".** A Figura 21 mostra o método **encrypt** com os parâmetros necessários para cifrar a mensagem com o algoritmo **AES**, modo de operação **CTR** e ***NoPadding***.

Figura 21: Método de criptografia AES

```
public static byte[] encrypt(String plainText, String encryptionKey) throws Exception {
    Cipher cipher = Cipher.getInstance("AES/CTR/NoPadding", "SunJCE");
   SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes("ISO-8859-1"), "AES");
   cipher.init(Cipher.ENCRYPT_MODE, key,new IvParameterSpec(IV.getBytes("ISO-8859-1")));
   return cipher.doFinal(plainText.getBytes("ISO-8859-1"));
}
```

Fonte: (do Autor)

Para que o *payload* cifrado seja integrado ao cabeçalho novamente, converte-se o mesmo para hexadecimal. O método de decifragem **decrypt** também foi implementado e está anexado ao código, porém não foi utilizado pois não faz parte do objetivo deste trabalho decifrar a mensagem.

- **Método 6: Anexa o cabeçalho original com o *payload* cifrado:**

Tendo o *payload* criptografado em hexadecimal, basta anexar o cabeçalho (header) original para que o pacote seja montado novamente. Essa junção é feita através do método **MergeData.**

O programa segue o mesmo fluxo para os próximos pacotes até chegar no último da lista, por fim o arquivo pode ser salvo. Todos os pacotes com *payload* criptografados são mostrados na tela para que seja possível comparar a *String* original e a cifrada. As telas serão apresentadas posteriormente na Seção de Testes e Resultados.

Como dispositivo principal deste trabalho, foi utilizado um computador tipo PC, com processador Intel® Core™ i7-3612QM CPU @ 2.10GHz, 16GB de memória RAM e um disco SSD com capacidade de até 240GB. Como sistema operacional, foi utilizado Windows 10 Professional x64. A IDE utilizada para desenvolver a ferramenta foi o NetBeans IDE 8.1.

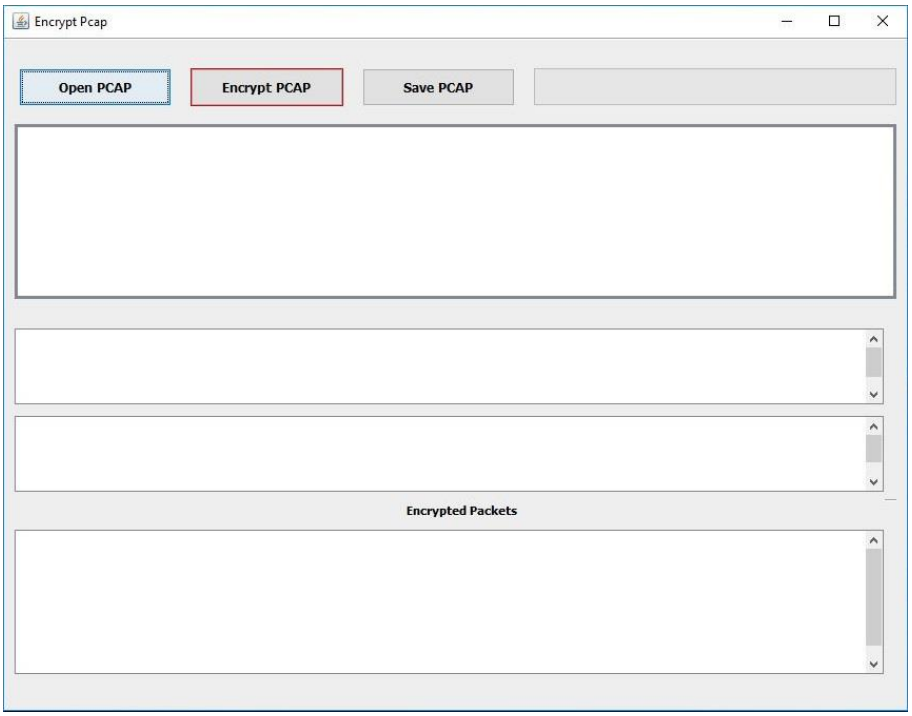
5 TESTES E RESULTADOS

Antes de utilizar a ferramenta desenvolvida, deve-se primeiro fazer a análise dos pacotes que contém ataques no *dataset*, já explicado na Seção 4.1 Análise dos pacotes maliciosos. Para a validação dos *datasets* cifrados foi utilizado o Wireshark, através dele foi possível autenticar que a criptografia foi aplicada no *payload* dos pacotes selecionados.

5.1 Metodologia para testes

Nesta Seção é apresentada a ferramenta desenvolvida para cifrar o *payload* dos pacotes maliciosos utilizando os *datasets* **testbed13junSMTPattack e hydra_telnet_only**. Também será detalhado o funcionamento do programa desde a abertura do arquivo *pcap* até a geração do *dataset* cifrado. O programa desenvolvido tem o nome de **Encrypt Pcap**, e para explicar seu funcionamento vamos primeiro analisar sua tela principal, ilustrada na Figura 22.

Figura 22: Tela principal do Encrypt Pcap

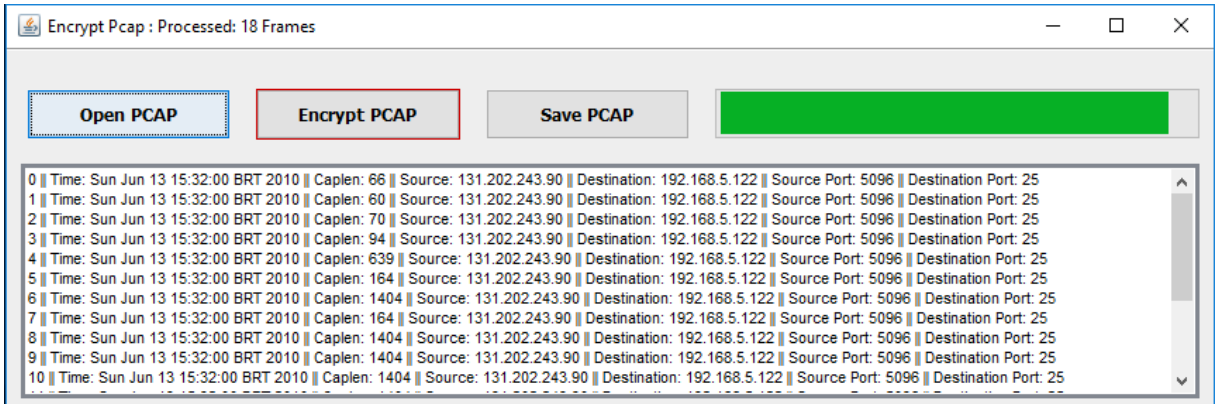


Fonte: (do Autor)

Com uma tela intuitiva, pode-se observar que o programa possui 3 botões e 4 painéis. O primeiro botão chamado **Open PCAP** faz a busca e leitura do arquivo (*dataset*) no sistema operacional, obrigatoriamente o usuário deve informar um arquivo com extensão ***pcap*** para que

a ferramenta consiga trabalhar com o mesmo. Após a abertura do arquivo, o programa lista todos os pacotes do *dataset* no primeiro painel que é semelhante ao Painel de Captura de Dados do Wireshark, conforme ilustrado na Figura 23.

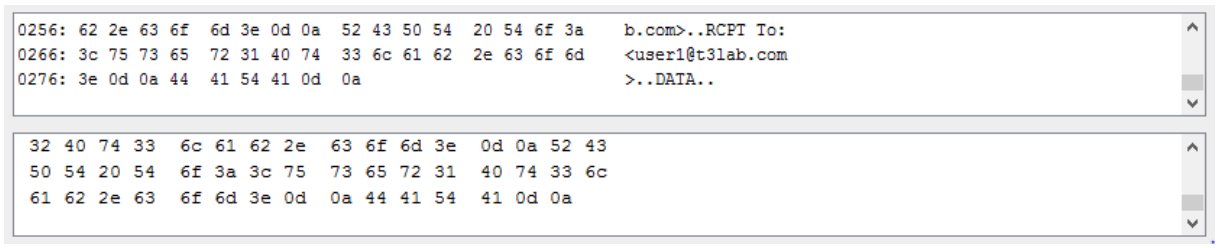
Figura 23: Painel que lista os pacotes do *dataset*



Fonte: (do Autor)

Ao selecionar um pacote dando dois cliques sobre ele, é possível observar que no segundo e terceiro painel são mostradas algumas informações importantes, como por exemplo, o cabeçalho e os dados em ASCII e também em hexadecimal. O PowerEditPcap em sua forma original possibilita que o usuário faça a edição dos valores hexadecimais no terceiro painel, porém este recurso não foi utilizado neste trabalho. Os dois painéis intermediários podem ser vistos na Figura 24.

Figura 24: Painéis de detalhes dos pacotes

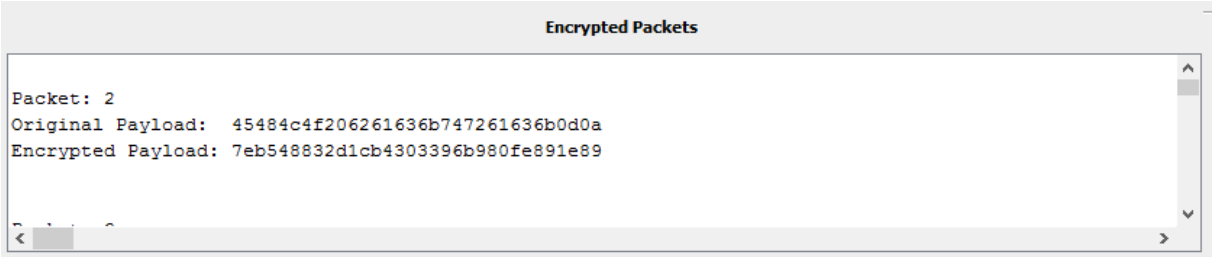


Fonte: (do Autor)

O segundo passo é clicar no botão **Encrypt PCAP**, e é neste momento que a criptografia é aplicada nos pacotes TCP, protocolos SMTP ou Telnet e que tenham a *flag* PSH setada com valor igual a 1, conforme definido no Método 2, detalhado na Seção 4.2. Quando a aplicação da cifragem for concluída, os pacotes cifrados serão mostrados no último painel chamado de

Encrypted Packets. Este mostra o número do pacote, o conteúdo de *payload* original e por fim o valor cifrado em hexadecimal, conforme representado pela Figura 25.

Figura 25: Painel Encrypted Packets



Fonte: (do Autor)

Para finalizar, basta clicar no botão **Save PCAP** e definir o local onde o novo *dataset* com pacotes cifrados deve ser salvo. Feito isso, a ferramenta pode ser encerrada e o arquivo pode ser buscado no sistema operacional para análise e validações.

Este capítulo apresentou as funcionalidades de cada botão da ferramenta e ilustrou os painéis com as informações que nela são exibidas. Os testes realizados para validar os *datasets* serão apontados na Seção 5.2.

5.2 Resultados Obtidos

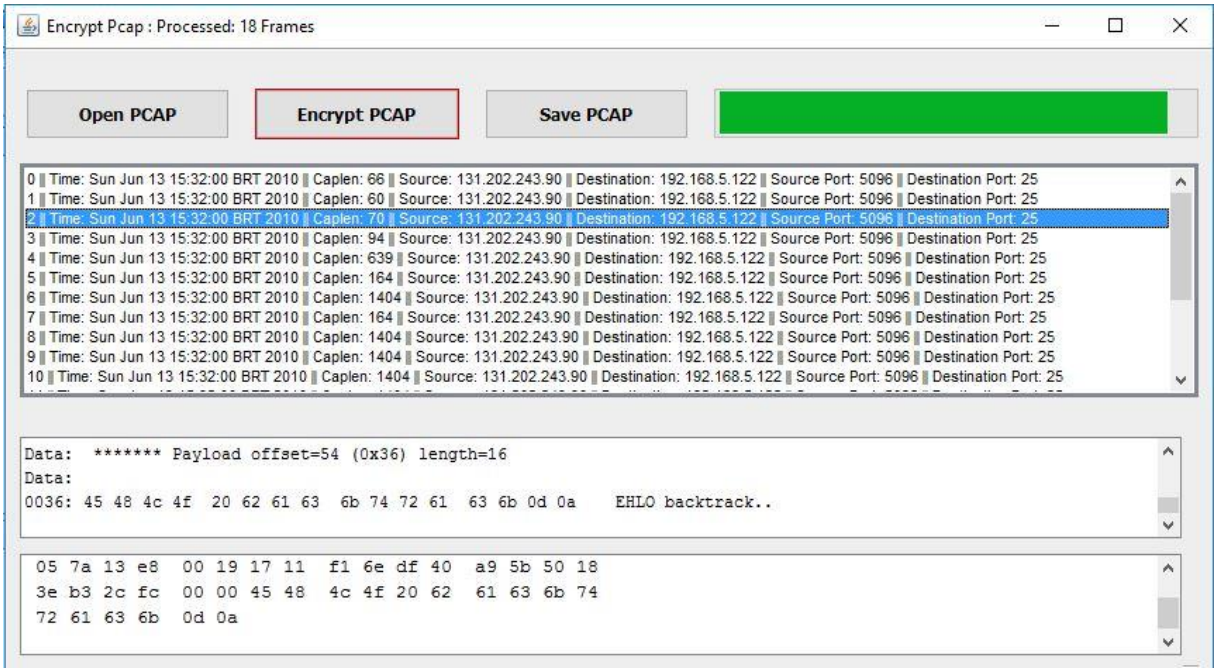
Os resultados obtidos através da ferramenta desenvolvida foram satisfatórios, pois tinha-se como objetivo principal cifrar o *payload* dos pacotes maliciosos sem modificar o cabeçalho, e isso foi possível utilizando os métodos de criptografia AES com modo de operação CTR. Para melhor entendimento, dividiu-se a apresentação dos resultados em dois cenários:

- Cenário 1: resultados obtidos com o *dataset* **testbed13junSMTPattack**
- Cenário 2: resultados obtidos com o *dataset* **hydra_telnet_only**

Cenário 1: *dataset* testbed13junSMTPattack

Este *dataset* contém um fluxo de pacotes com ataques introduzidos através do protocolo SMTP e foi gerado a partir do *dataset* ISCX IDS 2012 – dia 13 - disponibilizado pela Universidade de New Brinswick, no Canadá. A Figura 26 apresenta o terceiro pacote em sua forma original dentro da ferramenta **Encrypt Pcap** desenvolvida neste trabalho.

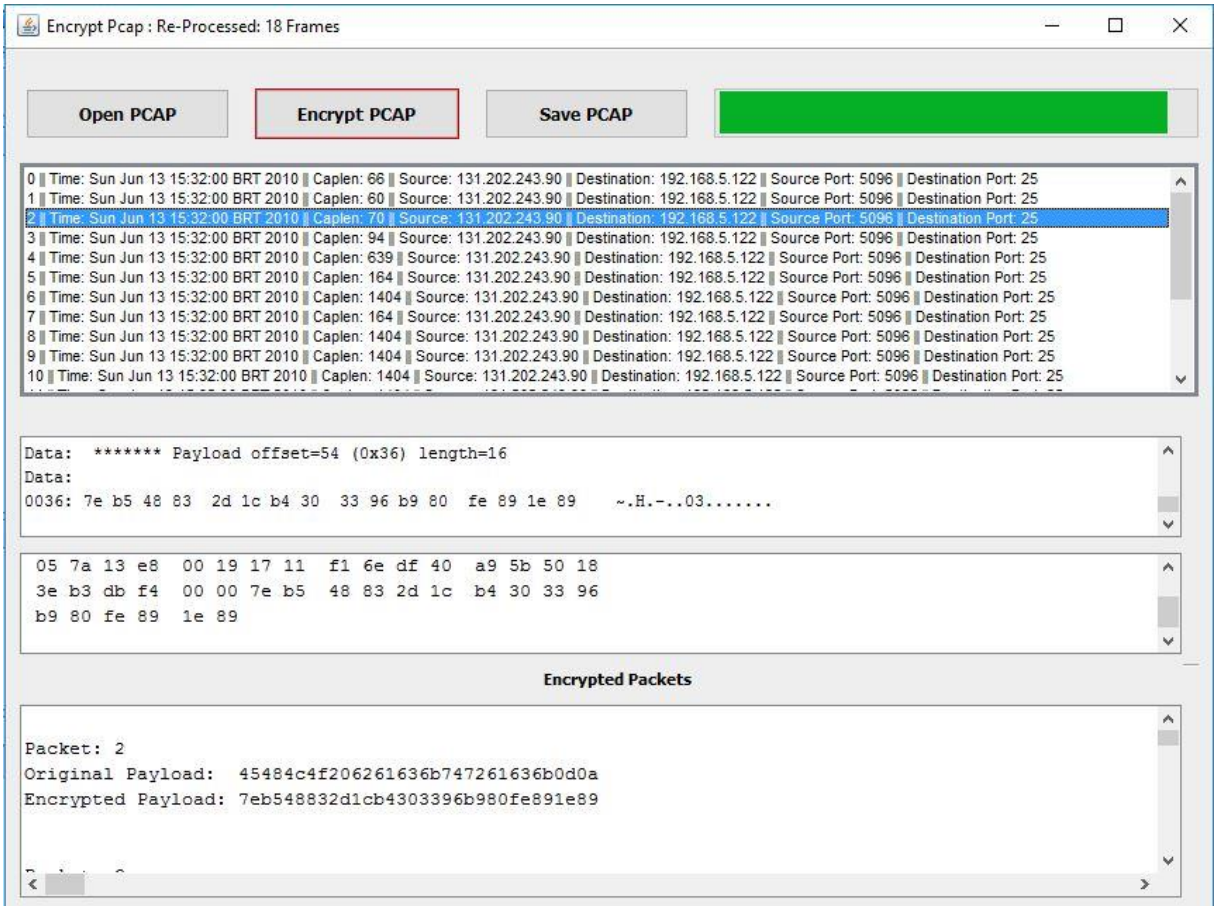
Figura 26: Terceiro pacote original



Fonte: (do Autor)

Então, aplicando a criptografia sobre o *dataset*, obtêm-se o resultado ilustrado na Figura 27.

Figura 27: Terceiro pacote cifrado

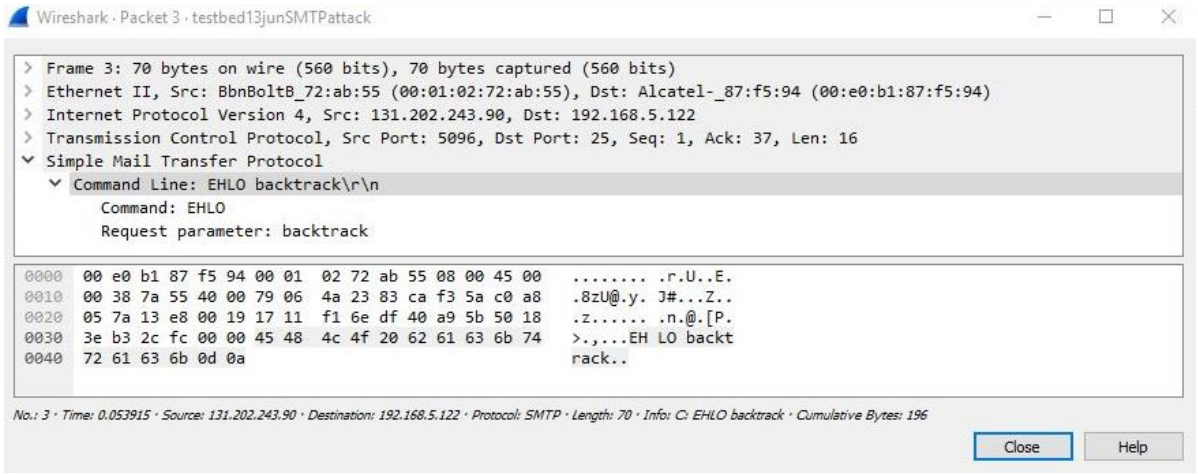


Fonte: (do Autor)

Analisando primeiramente o segundo painel da Figura 26, pode-se perceber que o conteúdo do *payload* representado por **EHLO backtrack** em sua forma original, teve alteração no mesmo painel da Figura 27. A informação foi criptografada gerou como resultado **~.H.-..03**, ou seja, em um conteúdo cifrado e incompreensível. Além disso, o painel **Encrypted Packets** indica mudanças no *payload* deste pacote, mostrando seu conteúdo original e pós aplicação da cifragem. Todos os outros pacotes do *dataset* que trazem os requisitos do Método 2 terão seus dados criptografados e apresentados neste painel.

Para validar o funcionamento do *dataset* cifrado, abrimos os dois arquivos (original e cifrado) no Wireshark a fim de comprovar que os valores do *payload* apresentados na ferramenta **Encrypt Pcap** realmente foram alterados. A Figura 28 mostra o mesmo pacote da Figura 26 dentro do Wireshark.

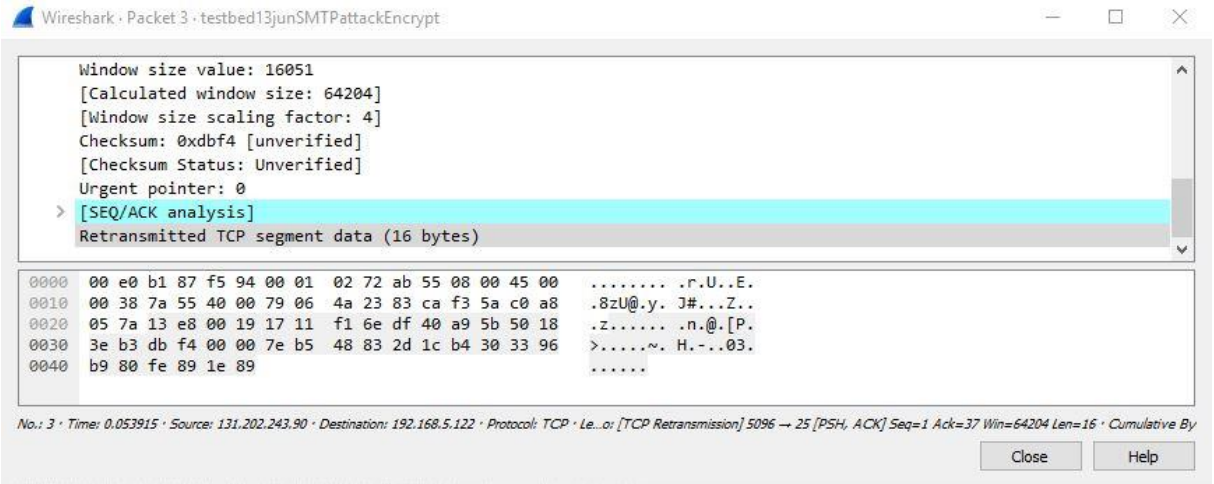
Figura 28: Terceiro pacote original no Wireshark



Fonte: (do Autor)

Em seguida, abrimos o arquivo que contém o pacote cifrado para comparar os valores hexadecimais e o texto plano. Na Figura 29 pode-se perceber que o conteúdo realmente foi alterado.

Figura 29: Terceiro pacote cifrado no Wireshark

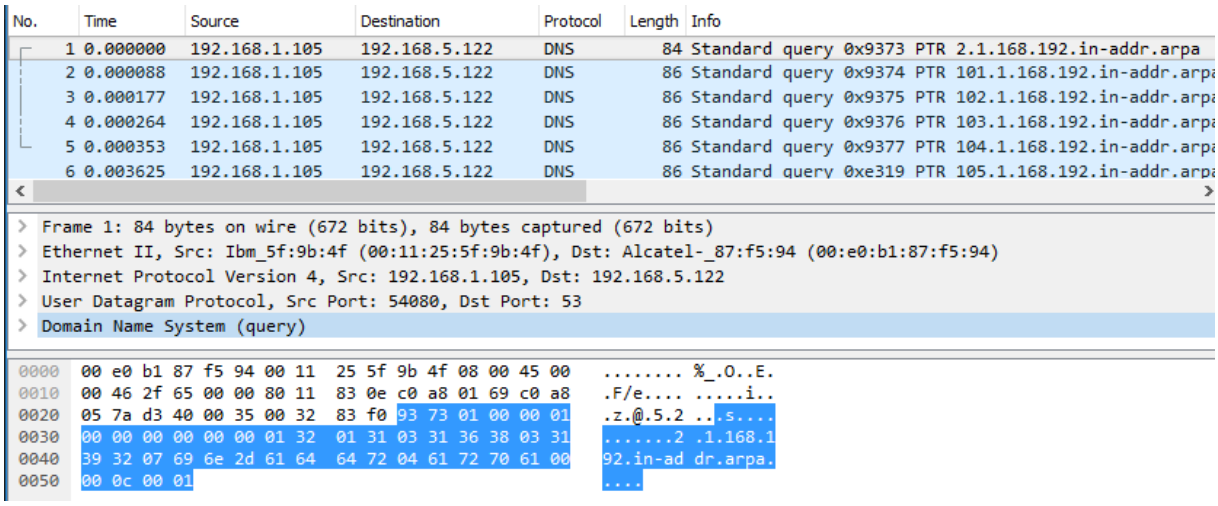


Fonte: (do Autor)

As imagens acima apresentadas comprovam que o *dataset* cifrado neste cenário teve seu conteúdo modificado pelos métodos de criptografia AES. Outros fluxos de ataques foram gerados a partir do *dataset* ISCX IDS 2012, porém alguns apresentaram anomalias após a aplicação da cifragem no *payload*, como por exemplo os pacotes maliciosos que utilizam o protocolo DNS. Estes tiveram seu conteúdo cifrado, porém exibem um alerta de **"Malformed Packet"**. Logo, descartou-se estes *datasets* pois não podemos garantir a integridade dos

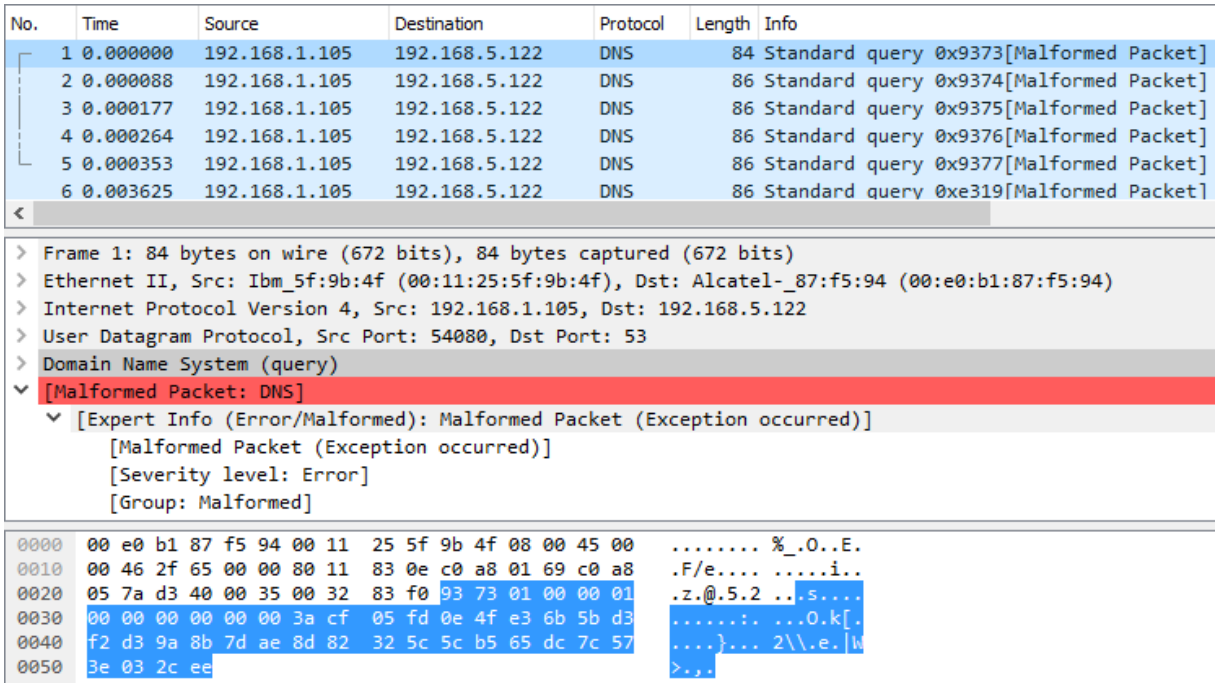
mesmos. A Figura 30 ilustra o pacote com ataques ao DNS em sua forma original, já a Figura 30 exibe o mesmo pacote com o *payload* cifrado mais o "erro" de pacote malformado.

Figura 30: Pacote original com ataques ao DNS



Fonte: (do Autor)

Figura 31: Pacote cifrado com ataques ao DNS



Fonte: (do Autor)

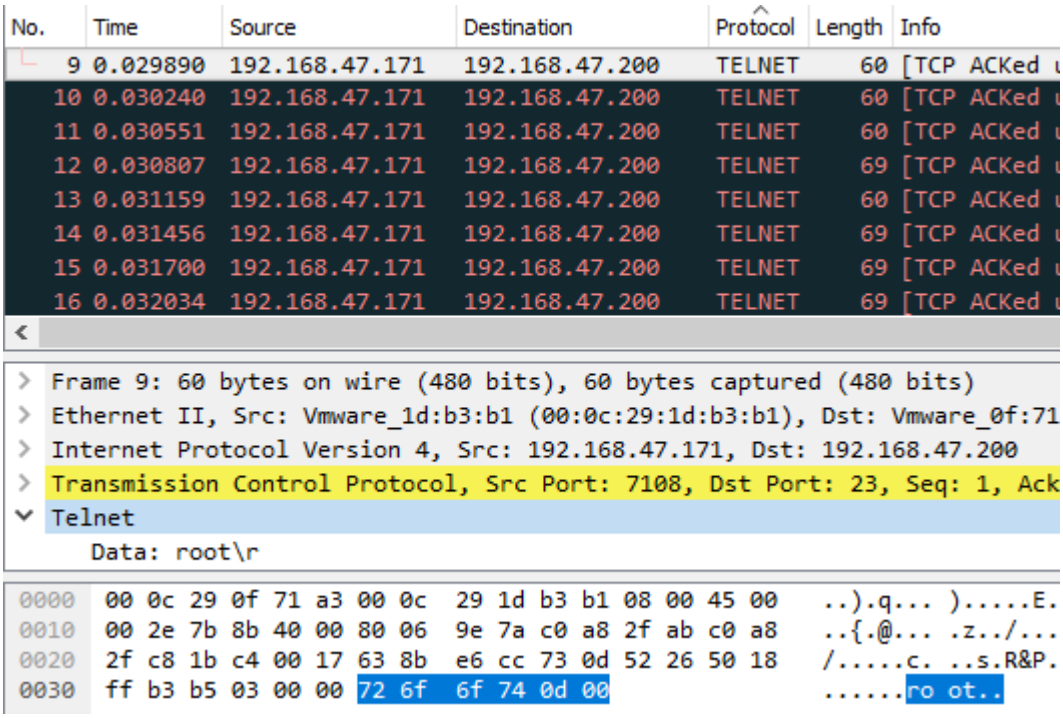
A ideia inicial deste trabalho de conclusão era cifrar todo o *dataset* ISCX IDS 2012 do dia 13, este contendo 5763149 pacotes totalizando em um arquivo de 3.95GB. Porém, a ferramenta desenvolvida não conseguiu abrir todo o arquivo, após alguns minutos rodando, a

memória da IDE foi excedida e um erro foi apresentado pelo NetBeans. Foram feitos testes chamando o programa Java com parâmetros de memória maiores, sistemas operacionais com arquitetura de 32 *bits*, e diversas modificações no código com o objetivo de solucionar o problema, mas nenhuma com sucesso. Por este motivo, decidiu-se trabalhar com fluxos de ataques menores exportados do arquivo original, pois desta forma o programa conseguiu trabalhar normalmente e com mais rapidez na cifragem.

Cenário 2: *dataset* hydra_telnet_only

No segundo cenário foi utilizado o *dataset* hydra_telnet_only, este contendo ataques de força bruta gerados pela ferramenta Hydra através do protocolo Telnet. Decidiu-se cifrar somente os pacotes que tem como destino a porta 23, caracterizando o ataque de *bruteforce*. A Figura 32 mostra uma sequência de pacotes com ataques direcionados para a porta 23 do serviço Telnet, evidenciando a intrusão.

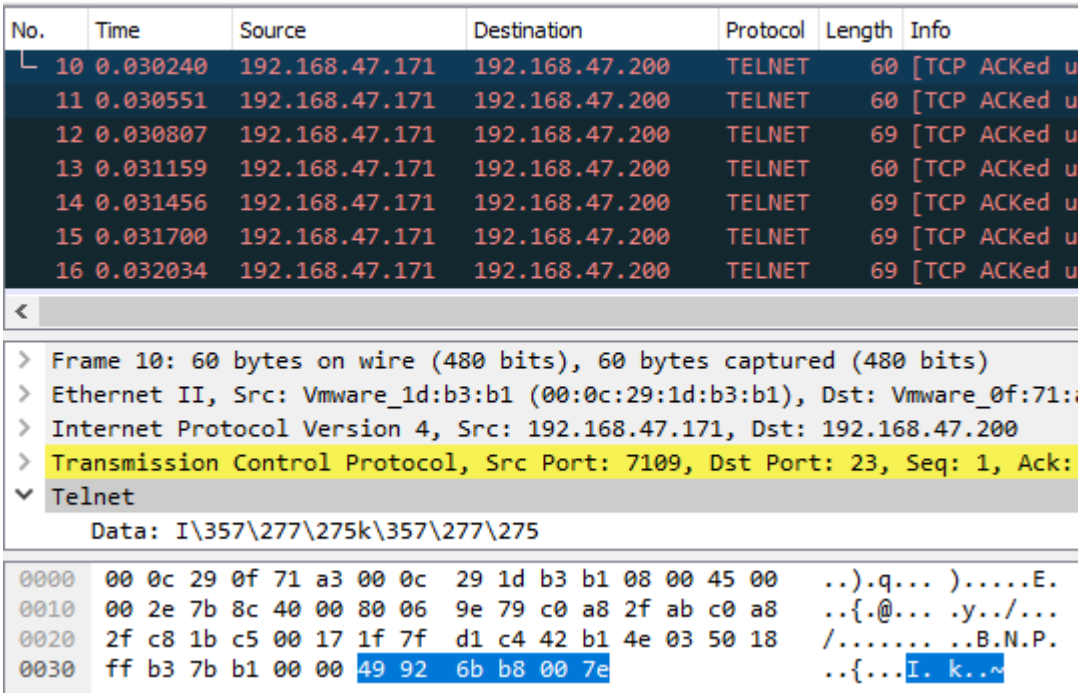
Figura 32: Pacote original com ataque bruteforce



Fonte: (do Autor)

Utilizando a ferramenta **Encrypt Pcap** desenvolvida para cifrar o *payload* dos pacotes, obteve-se o resultado representado pela Figura 33.

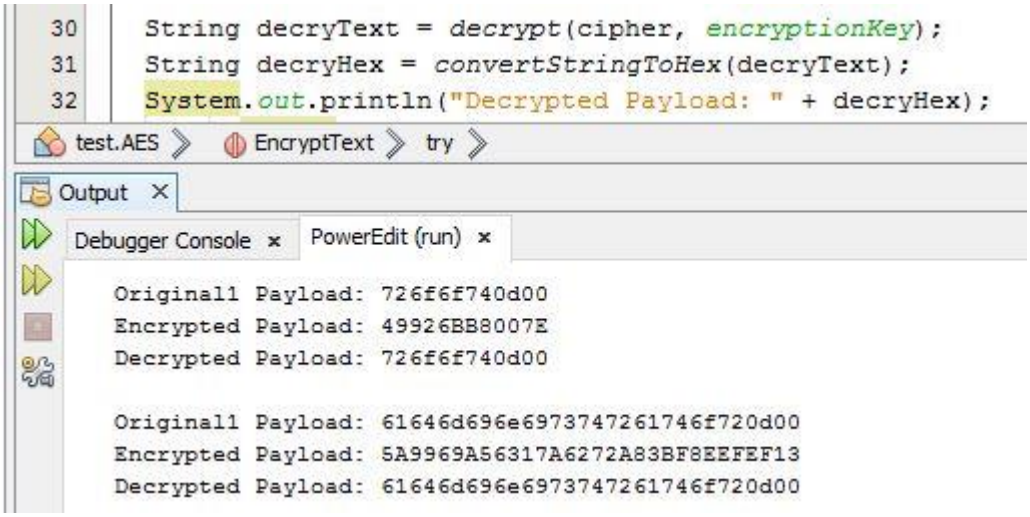
Figura 33: Pacote cifrado com ataque bruteforce



Fonte: (do Autor)

O *dataset* gerado neste cenário agora contém ataques cifrados de *bruteforce* em sequência. Com o objetivo de comprovar a correto funcionamento do método de decifragem, decidiu-se imprimir no console do NetBeans dois pacotes deste *dataset*, onde exibe-se o *payload* em sua forma original (em Hexadecimal), em seguida o *payload* cifrado (em Hexadecimal) e por fim, o *payload* voltando a sua forma original aplicando a descriptografia através do método **decrypt,** utilizando a mesma chave inserida para criptografar anteriormente. A Figura 34 ilustra o método juntamente com os resultados impressos no console.

Figura 34: Método de decifragem e resultados no console



Fonte: (do Autor)

Finalizando, é possível afirmar que cifrando o *payload* dos pacotes através da ferramenta desenvolvida, se manteve a integridade dos pacotes possibilitando a execução e análise do novo arquivo *pcap* no Wireshark mesmo após alterar seu conteúdo original.

Se comparado a outros *datasets* como o de Sperotto *et al.* que fornece uma base de dados com tráfego malicioso para detecção de intrusão baseada em fluxo, este trabalho de conclusão de curso traz um grande diferencial que são os ataques cifrados, levando-se em consideração o aumento significativo de ataques no tráfego criptografado e a deficiência dos IDSs em identificar estes ataques.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Atualmente, as bases de dados tradicionais existentes, como a KDDCUP99 e NSL-KDD não possuem ataques criptografados. Muitos conjuntos de dados, utilizados para avaliação de IDS, são internos e não podem ser compartilhados por questões de privacidade, outros sofreram um processo para tornar anônimos os usuários ou ainda foram processados e certos dados foram removidos, alterando características importantes do conjunto e assim não refletem o verdadeiro comportamento na rede (SHIRAVI, 2012).

Com a evolução da tecnologia, a criptografia é algo amplamente utilizado, tanto em aplicações comerciais quanto domésticas, pois os ataques podem comprometer a segurança da informação e se proteger contra isso é extremamente importante. Assim como a criptografia pode ser utilizada para segurança, ela também pode ser usada para facilitar a entrada de invasores, pois um atacante pode empregar criptografia com o objetivo de mascarar seus ataques.

Neste trabalho de conclusão foi apresentado uma abordagem de como criar um *dataset* com ataques criptografados. Estes *datasets* são importantes para testar métodos de detecção de intrusão. O trabalho também tem como propósito contribuir com trabalhos futuros na área de detecção de ataques criptografados.

Após a etapa de implementação da ferramenta de cifragem, foram testados dois *datasets* com diferentes tipos de ataques, e concluiu-se que o objetivo foi atingido, através do Wireshark pode-se validar que o *payload* dos pacotes com ataques foi criptografado. O algoritmo AES se mostrou eficaz quando utilizado com o modo de operação CTR, cifrando os valores do texto e mantendo o tamanho original do pacote.

Como melhorias possíveis para este trabalho, pode-se mencionar a otimização do código na tentativa de utilizar menos recursos de memória. Outro ponto que pode ser melhorado é a escolha dos protocolos que o usuário deseja cifrar através de interface gráfica, pois hoje essa indicação é feita via código. Com a ferramenta desenvolvida, pesquisadores podem cifrar outros *datasets* com ataques diversificados, disponibilizando assim um acervo maior de bases de dados para testar novos métodos de detecção de intrusão criptografada.

REFERÊNCIAS

BURNETT, S.; PAINE, S. – *Criptografia e Segurança* – O guia oficial RSA. 1ª edição. Rio de Janeiro: Campus, 2002.

DEBAR, H., DACIER, M. and WESPI, A. *A Revised Taxonomy for Intrusion Detection Systems*. Annals of Telecommunications, 2000. p. 361-378.

DE CAMPOS, LIDIO MAURO LIMA; LIMA, ALBERTO SAMPAIO. *"Network intrusion detection system using data mining." Internacional Conference on Engineering Applications of Neural Networks.* Springer Berlin Heidelberg, 2012. p. 104-113.

EID, H. F. *et al*. *Security Technology*, v. 259. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 195–203.

FAGUNDES, BRUNNO; NEU, CHARLES V.; OROZCO, ALEX M. S.; MICHELIN, REGIO A.; ZORZO, AVELINO F. Snortik – *Uma integração do IDS SNORT e o sistema de firewall do MICROTIK ROUTEROS*. 14ª Escola Regional de Redes de Computadores, 27–30 de setembro de 2016, Porto Alegre – RS, editora: Sociedade Brasileira de Computação.

FOROUSHANI, V.A.; ADIBNIA, F.; HOJATI, E. – "Intrusion detection in encrypted accesses with ssh protocol to network public servers" *Computer and Communication Engineering. ICCCE 2008. International Conference on,* 2008. vol. 52, n. 98, p. 314-318.

FOUNDATION, Open Networking. Open Networking Foundation white papers. Disponível em: <https://www.opennetworking.org/sdn-resources/sdn-definition>. Acesso em: Maio de 2017.

GARCIA, THIAGO O. *Definição de novas regras para o IDS Snort em redes definidas por software*. 2016. 54 f. Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação) - Universidade de Santa Cruz do Sul – UNISC, Santa Cruz do Sul, 2016.

GILMORE, J. – Enterpreneur and Civil Libertarian. Disponível em: <http://www.toad.com/gnu/>. Acesso em: Maio de 2017

GODOI, EVERTON. *Detecção de intrusos (IDS), conceitos e implantação do SNORT*; 2007. Disponível em: <https://www.vivaolinux.com.br/artigo/Deteccao-de-intrusos-(IDS)-conceitos-e-implantacao-do-SNORT?pagina=2>. Acesso em: Outubro de 2016.

GOGOI, PRASANTA el al. Packet and flow based network intrusion dataset. Em: International Conference on Contemporany Computing. Springer Berlin Heidelberg, 2012. p. 322-334.

HE, G.; ZHANG, T.; XU, B. – "A novel method to detect encrypted data exfiltration" *in Advanced Cloud and Big Data (CBD), Second International Conference on,* 2014. p. 240-246.

HUANG, L.; ZHI, X.; GAO, Q.; KAUSAR, S.; ZHENG, S. – "*Design and implementation of multicast routing system over snd and sflow*" in 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN). IEEE Conference Publications, 2016. p. 524-529.

JANKOWSKI, D.; AMANOWICZ, M. – "Intrusion detection in software defined networks with self-organized maps" *Journal of Telecommunications & Information Technology,* 2015. p. 3-9.

JNETPCAP. *Sly Technologies jNetPcap.* Disponível em: < http://jnetpcap.com/>. Acesso em: Junho de 2017.

JOZWIAK, I.; KEDZIORA, M.; MELINSKA, A.: *Theoretical and Practical Aspects of Encrypted Containers Detection* – Digital Forensics Approach. Dependable Computer Systems, 2011. vol. 97, p. 75-85.

KDDCUP99. *KDD Cup 1999 Data*. Disponível em: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>. Outubro, 1999. Acesso em: Agosto de 2016.

KENDALL, KRISTOPHER. *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*. MASSACHUSETTS INST OF TECH CAMBRIDGE DEPT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, 1999.

KOCH, R.; GOLLING, M.; RODOSEK, G. – "Behavior-based intrusion detection in encrypted environments" *IEEE Communications Magazine, IEEE,* 2014. vol. 52, n. 14450733, p. 124-131.

KUROSE, JAMES F.; ROSS, KEITH W. *Rede de Computadores e a Internet*: uma abordagem top-down. [S.1.]: Pearson Education, 2010.

LIPPMANN, R. P. *et al.*, "*Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation*," *discex,* 2000. vol. 02, p. 1012.

LYDA, ROBERT; HAMROCK, JAMES. *Using Entropy Analysis to Find Encrypted and Packed Malware*. IEEE COMPUTER SOCIETY, 540-7993/07, 2007.

MORENO, E; PEREIRA, F. D.; CHIARAMONTE, R.B. – *Criptografia em Software e Hardware*; Novatec; 2005.

MOTA FILHO, JOÃO ERIBERTO. *Análise de tráfego em redes TCP/IP: utilize tcpdump na análise de tráfego em qualquer sistema operacional;* Novatec; 2013.

MZILA, P.; DUBE, E. The effect of destination linked feature selection in real-time network intrusion detection. Em ICIMP 2013: *8th International Conference on Internet Monitoring and Protection.* 2013.

NETBEANS. *NetBeans IDE.* Disponível em: <https://netbeans.org/>. Acesso em: Junho de 2017.

NEU, CHARLES V.; ZORZO, AVELINO F.; OROZCO, ALEX M.S.; MICHELIN, REGIO A. – *An approach for detecting encrypted insider attacks on OpenFlow SDN Networks*. 11th International Conference for Internet Technology and Secured Transactions, Barcelona, Espanha, publicado pela IEEEXPLORE. 2016.

NSL-KDD. *The NSL-KDD Data Set*. Disponível em: < http://www.unb.ca/research/iscx/dataset/iscx-NSL-KDD-dataset.html>. Acesso em: Outubro de 2016.

OPENFLOW, "*Openflow switch specification, version 1.0.*0" Disponível em: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf>. Acesso em: Novembro de 2016.

SALLAY, H.; AMMAR, A.; BEN SAAD, M.; BOUROUIS, S. "*A real time adaptive intrusion detection alert classifier for high speed networks*. Em Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on, 2013. p. 73-80.

SHERRY, J.; LAN, C.; POPA, R.A.; RATNASAMY, S. – "Blindbox: Deep packet inspection over encrypted traffic" *SIGCOMM Comput. Commun. Rev.*, 2015. vol. 45, n. 4, p. 213-226.

SHIRAVI, A.; SHIRAVI, H.; TAVALLAEE, M.; GHORBANI, A. A. "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, 2012. v. 31, n. 3, p. 357-374.

SILVA, RENATO MAIA. *Redes Neurais Artificiais aplicadas à Detecção de Intrusão em Redes TCP/IP*. 2005. 144 f. Dissertação (Mestrado) – PUC-RIO – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro. 2005.

SINGH, ABHINAV. *Instant Wireshark Starter.* Packt Publishing Ltd, 2013.

SNORT. *The Snort Project*. Disponível em: <http://www.snort.org/>. Acesso em: Novembro de 2016.

SOURCEFORGE. *PowerEdit-Pcap.* Disponível em: <https://sourceforge.net/projects/powereditpcap/>. Acesso em: Junho de 2017.

SPEROTTO, ANNA; SADRE, RAMIN; VLIET, FRANK VAN; PRAS, AIKO. A labeled data set for flow-based intrusion detection. In International Workshop on IP Operations and Management. Springer Berlin Heidelberg, 2009. p. 39-50.

STALLINGS, W. Criptografia e Segurança de Redes – 6ª edição. São Paulo: Pearson Education, 2014.

SYMANTEC. *What you need to know about the WannaCry Ransomware*. Disponível em: <https://www.symantec.com/connect/blogs/what-you-need-know-about-wannacry-ransomware>. Acesso em: Julho de 2017.

TANENBAUM, A. S. - *Redes de Computadores*. Campus; 4ª edição; 2003.

TAVALLAEE, M.; BAGHERI, E.; LU, W.; GHORBANI; A. A. *A Detailed Analysis of the KDD CUP 99 Data Set*. Proceeding of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications. (CISDA), 2009.

TAVALLAEE, M. STAKHANOVA, N. GHORBANI, A. A. “Toward Credible Evaluation of Anomaly-Based Intrusion-Detection Methods,” *IEEE Trans. Syst. Man, Cybern. Part C Applications Rev.*, 2010. v. 40, n. 5, p. 516–524.

TCPDUMP. *TCPDUMP/LIBCAP public repository*. Disponível em: <http://www.tcpdump.org>. Acesso em: Outubro de 2016.

TCPTRACE. OSTERMAN, S. Disponível em: <http://www.tcptrace.org>. Acesso em: Outubro de 2016.

THC.org. *The Hacker´s Choice.* Disponível em: <http://www.thc.org>. Acesso em: Junho de 2017.

THURNER, SIMON; GRUN, MARCEL; SCHMITT, SVEN; BAIER, HARALD. Improving the detection of encrypted data on storage devices. *Ninth International Conference on IT Security Incident Management & IT Forensics.* 2015.

TSHARK. *Tshark – Dump and analyze network traffic*. Disponível em: <https://www.wireshark.org/docs/man-pages/tshark.html>. Acesso em: Novembro de 2016.
UCHÔA, J. Q. – *Segurança em Redes e Criptografia;* UFLA; Lavras – MG; 2003.

UNB. *Faculty of Computer Science*. University of New Brunswick. Canada. Disponível em: <http://www.unb.ca/research/iscx/dataset/iscx-IDS-dataset.html>. Acesso em: Novembro de 2016.

VELOSO, CAIO J. MARTINS. *Criptologia* – Uma ciência fundamental para tratamento de informações sigilosas, 2002. Disponível em: <http://www. modulo.com.br>. Acesso em: Outubro de 2016.

WEKA. *Weka 3 – Data Mining with Open Source Machine Learning Software in Java*. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/>. Acesso em: Outubro de 2016.

WIRESHARK. *Wireshark – Go Deep.* Disponível em: <http://www.wireshark.org>. Acesso em: Novembro de 2016.

ZHANG, H; PAPADOPOULOS, C.; MASSEY, D.: *Detecting Encrypted Botnet Traffic.* Computer Science Department, Colorado State University, 2003.

ZHONG, S. H., HUANG, H. J. and BIN CHEN, A. “An Effective Intrusion Detection Model Based on Random Forest and Neural Networks,” *Adv. Mater. Res.*, 2011. v. 267, p. 308–313.

Santa Cruz do Sul, 22 de junho de 2017.

_____

Igor Henrique Martini

Aluno

_____

Professor Me. Charles Varlei Neu

Orientador