UNIVERSIDADE FEDERAL DE SANTA MARIA
COLÉGIO TÉCNICO INDUSTRIAL DE SANTA MARIA
CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMPUTADORES

JONATHAN ORTIZ PREUSS

DESENVOLVIMENTO DE FERRAMENTA PARA INTEGRAÇÃO E CONTROLE DE ACESSO PARA FIREWALL

Santa Maria, RS

CTISM/UFSM,RS ORTIZ PREUSS, Jonthan Tecnólogo

JONATHAN ORTIZ PREUSS

DESENVOLVIMENTO DE FERRAMENTA PARA INTEGRAÇÃO E CONTROLE DE ACESSO PARA FIREWALL

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Redes de Computadores da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de **Tecnólogo em Redes de Computadores**

Orientador: Prof. Me. (UFSM) Renato Preigschadt de Azevedo

Co-orientador: Prof. Tecg. (UFSM) Bolivar Menezes da Silva

Santa Maria, RS

ORTIZ PREUSS, JONATHAN

DESENVOLVIMENTO DE FERRAMENTA PARA INTEGRAÇÃO E CONTROLE DE ACESSO PARA FIREWALL / por JONATHAN ORTIZ PREUSS. – .

43 f.: il.; 30 cm.

Orientador: Renato Preigschadt de Azevedo

Co-orientador: Bolivar Menezes da Silva

Trabalho de Conclusão de Curso - Universidade Federal de Santa Maria, Colégio Técnico Industrial de Santa Maria, Curso Superior de Tecnologia em Redes de Computadores, RS, .

1. Firewall. 2. Segurança. 3. Redes. 4. Desenvolvimento. I. Preigschadt de Azevedo, Renato. II. Menezes da Silva, Bolivar. III. Título.

E-mail: autor@email.com

JONATHAN ORTIZ PREUSS

DESENVOLVIMENTO DE FERRAMENTA PARA INTEGRAÇÃO E CONTROLE DE ACESSO PARA FIREWALL

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Redes de Computadores da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de **Tecnólogo em Redes de Computadores**

Aprovado em de de :

Renato Preigschadt de Azevedo, Me. (UFSM)
(Presidente/Orientador)

Tarcisio Ceolin Junior, Me (UFSM)

Rafael Teodósio Pereira, Phd (UMINHO)

Santa Maria, RS

DEDICATÓRIA

Gostaria de dedicar a minha família e a todos que contribuíram de alguma forma até esse momento.

AGRADECIMENTOS

A realização desse trabalho não seria possível sem o apoio dos professores Renato P. Azevedo e Bolivar Menezes da Silva.

“Domine seu medo ou vire refém dele.”
(AUTOR DESCONHECIDO)

RESUMO

DESENVOLVIMENTO DE FERRAMENTA PARA INTEGRAÇÃO E CONTROLE DE ACESSO PARA FIREWALL

AUTOR: JONATHAN ORTIZ PREUSS
ORIENTADOR: RENATO PREIGSCHADT DE AZEVEDO
CO-ORIENTADOR: BOLIVAR MENEZES DA SILVA

O uso crescente da rede mundial de computadores nos últimos anos trouxe um enorme leque de serviços e facilidades, tanto para usuários como pra empresas e organizações. Empresas de grande porte acabam por usar a internet como uma ferramenta para transmitir dados comerciais e sigilosos entre suas filiais. Neste sentido a segurança dos dados trafegados e a integridade da rede se torna um fator crucial nesse cenário. Normalmente redes de grande porte, possuem um *firewall* responsável pela segurança da comunicação, e esse dispositivo tem sua gerência centralizada em uma equipe de administradores, tornando assim centralizado o poder de ação mediante a um incidente de segurança. Este trabalho visa criar uma solução que permite a ação de administradores de segmento de uma forma contida na configuração do *firewall*, adequando as configurações do mesmo às necessidades do setor ao qual esse administradores local pertence.

Palavras-chave: Firewall. Segurança. Redes. Desenvolvimento.

ABSTRACT

DEVELOPMENT OF TOOL FOR INTEGRATION AND ACCESS CONTROL FOR FIREWALL

AUTHOR: JONATHAN ORTIZ PREUSS
ADVISOR: RENATO PREIGSCHADT DE AZEVEDO
COADVISOR: BOLIVAR MENEZES DA SILVA

The increasing use of the World Wide Web in recent years brought a huge range of services and facilities for the users, for companies and organizations. The big companies end up using the internet as a tool to transmit commercial and sensitive data between its subsidiaries. The security of the transmitted data and the network integrity becomes a crucial factor in this scenario. Usually big networks have a firewall responsible for the security of communication, this device is managed by a single administrator, making centralized the power to act in a security incident. This work aims to create a solution that allows the action of segment administrators in a way contained in the configuration of the firewall, adapting and adjusting your configuration according with needs of the sector to which these local administrators belongs.

Keywords: Firewall. Security. Network. Development.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ICMP	Internet Control Message Protocol
SSL	Secure Sockets Layer
ER	Entidade relacionamento
CTISM	Colégio Técnico Industrial de Santa Maria
CT	Centro de Tecnologia
UFSM	Universidade Federal de Santa Maria
CIDR	Classless Inter-Domain Routing
IP	Internet Protocol
XML	eXtensible Markup Language
HTTP	Hyper Text Transfer Protocol
TCP	Transmission Control Protocol
VPN	Virtual Private Network
CDN	Content Delivery Network
DDoS	Distributed Denial of Service
NAT	Network address translation
SSL	Secure Socket Layer
SQL	Structured Query Language

SUMÁRIO

1 INTRODUÇÃO

Com a crescente expansão da rede mundial de computadores nas últimas décadas e o seu emprego em âmbito empresarial e institucional, fornecendo facilidades como: a agilidade no envio e recebimento de informações bem como o compartilhamento de recursos, essas facilidades tendem a tornar algumas atividades de uma empresa dependente da disponibilidade e funcionalidade da rede. A indisponibilidade ou a presença de um comportamento anômalo do serviço de rede pode ocasionar impedimento na execução das tarefas de uma empresa e dessa forma acarretar em prejuízos operacionais, financeiros e também a integridade dos dados da empresa.

Em paralelo a esse crescimento e as facilidades oferecidas ocorre um aumento também nos incidentes de segurança. Segundo (AKAMAI, 2016) em relação ao primeiro trimestre de 2016, houve um aumento de 9% nos ataques do tipo DDoS totais e 197% no ataque de aplicação WEB provenientes do Brasil, deixando assim o Brasil no topo da lista dos países originários de ataques WEB.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

A presente monografia tem como objetivo geral o desenvolvimento de uma ferramenta e um método para a administração remota de regras em *firewalls*. A ferramenta desenvolvida tornará possível que usuários sem permissão administrativa do dispositivo que hospeda um *firewall* de borda possam em um ambiente supervisionado e controlado, operar novas regras no *firewall*.

1.1.2 Objetivos Específicos

- Efetuar uma revisão bibliográfica sobre trabalhos já implementados;
- Pesquisar tecnologias para elaborar a ferramenta proposta;
- Desenvolver a ferramenta para interagir com os dispositivos de *firewall*;
- Realizar testes e experimentos práticos usando o método e aplicação desenvolvida para

verificar suas competências;

- Coletar, analisar e publicar os resultados obtidos;

1.2 JUSTIFICATIVA

Empresas e entidades de pequeno a grande porte, que possuem filiais ou instalações geograficamente distribuídas, frequentemente deparam-se como a necessidade de interligar sua infraestrutura de redes para o compartilhamento de recursos ou dados corporativos, para esse cenário normalmente é adotado o uso de VPNs (Virtual Private Networks). A utilização desse serviço fornece uma solução de tunelamento através da internet, com um baixo custo e um certo nível de segurança. Nessa conjuntura também é comum que, cada filial possua um *firewall* e administradores próprios para a gerência da rede local e a matriz possua um *firewall* de borda. Habitualmente, esses administradores de redes não possuem acesso ou privilégio suficiente para operar dispositivos de *firewall* que estejam fora dos seus domínios administrativos. A implementação do projeto proposto tornará possível que administradores, que tenham sub-redes internas gerenciadas por diferentes administradores de segmentos possam fornecer um acesso, de forma controlada, para que esses administradores locais tenham acesso somente a suas referentes sub-redes. Nesse contexto esses administradores de segmentos serão capazes de aplicar no *firewall* de borda, modificações que impactarão somente nas suas sub-redes. Logo, diante de um comportamento anômalo na rede, torna-se possível que diferentes administradores possam reagir de forma a tomar medidas corretivas visando o aumento de confiabilidade, segurança e integridade de uma rede.

2 REVISÃO BIBLIOGRÁFICA

Esse capítulo apresenta conceitos e definições referentes a temas abordados ao longo do desenvolvimento da aplicação proposta, como conceitos de *firewall*, sockets e criptografia, Softwares de firewall e trabalhos relacionados a proposta dessa monografia.

2.1 *FIREWALL*

Firewall é um componente ou conjunto de componentes que restringe o acesso entre uma rede protegida e a Internet ou entre conjuntos de redes (ELIZABETH D. ZWICKY, 2000). O dispositivo atuante como *firewall* é responsável por analisar o tráfego de entrada e saída em uma rede e é baseado em uma série de políticas de controle de acesso o que permite ou não um determinado tráfego. Para fins de registro histórico do acionamento dessas políticas é possível armazenar o resultado da execução das mesmas em um arquivo de log[1]. Existem três principais tipos de *firewall*: filtro de pacotes tradicionais, filtros de estado e gateway de aplicação (JAMES F. KUROSE, 2016).

- Filtro de pacotes tradicionais: baseado em regras estabelecidas pelo administrador da rede, esse *firewall* analisa e descarta ou não os pacotes que entram e saem da mesma, normalmente usados para bloquear pacotes provenientes de endereços e serviços específicos.

- Filtros de estado: O filtro de estado opera em dois estágios, o tratamento inicial dos pacotes é semelhante ao tratamento realizado por um filtro de pacotes. Utilizando um conjunto de regras estipuladas pelo administrador da rede, o *firewall* irá analisar se um determinado pacote pode ou não trafegar na rede. Caso o tráfego seja permitido o *firewall* irá manter e vistoriar uma tabela com informações sobre o estado da conexão desse tráfego. Segundo (STEPHEN NORTHCUTT LENNY ZELTSER, 2005) o protocolo TCP, por ser orientado a conexão, é um bom exemplo para demonstrar o funcionamento de uma tabela de estados. Baseado nas flags de que sinalizam a comunicação do TCP, ao iniciar uma conexão o dispositivo insere na tabela de estado um registro referente a conexão em questão. O registro na tabela de estados contém: protocolo da comunicação, endereço de origem e destino, porta de origem e destino, estado da conexão e uma relação das portas de origem e destino inversas para demonstrar o tráfego de resposta.

[1] Log é o arquivo aonde são armazenados os registros dos eventos de um sistema.

Para protocolos como UDP que não são orientados a conexão, no momento em que a comunicação passa pelo conjunto de regras definidas pelo administrador, o dispositivo de stateful insere um registro na tabela de estado contendo os endereços de origem e destino da conexão, porta de origem e destino, estado da conexão e uma relação das portas de origem e destino inversas para demonstrar o tráfego de resposta e executa módulos para acompanhamento da conexão.

- Gateway de aplicação: trabalha com serviços mais específicos da camada de aplicação. Além da análise feita sobre IP e porta, os pacotes têm suas informações analisadas a nível de : o conteúdo, tamanho e protocolo averiguados. Um exemplo seria um *firewall* que trabalhe junto a um servidor web, filtrando o tráfego HTTP.

A aplicação Netfilter/Iptables é um *firewall* do tipo filtro de pacotes *stateful*. A qualificação de *stateful* é dada pra o *firewall* que possua métodos para reconhecer conexões, dessa forma é possível filtrar pacotes a partir do cenário de uma sessão e/ou conexão (ELIZABETH D. ZWICKY, 2000). O filtro analisa o cabeçalho dos pacotes verificando os endereços de origem e destino e, através de um conjunto de regras determinadas previamente, descarta ou encaminha o pacote até seu destino.

2.1.1 Netfilter

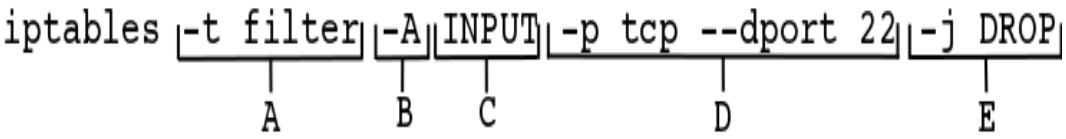
Netfiler é um framework para manipulação e filtro de pacotes, implementado no kernel Linux desde as versões 2.4.x em diante. O projeto Netfilter foi fundado por Paul Rusty Russell e é o sucessor dos subsistemas *ipchains* e *ipfwadm* presentes nas versões 2.2.x e 2.0.x, respectivamente. O Netfilter foi desenvolvido desde o início com o intuito de trazer uma forma mais descomplicada para trabalhar sobre a pilha de protocolos de rede presente no kernel Linux. O framework trata o fluxo de pacotes em forma de tabelas (quatro ao todo), sendo elas: tabela *filter*, tabela *raw*, tabela *nat* e tabela *mangle*. Cada tabela mantém situações de fluxo que são analisadas no momento de tratar o tráfego de dados. Essas tabelas são compostas por *chains*, sendo elas: *INPUT*, *OUTPUT*, *FORWARD*, *PREROUTIN*, *POSTROUTING*, onde cada *chain* contém um grupo de regras e uma política padrão (*policy*). Existem ações associadas a cada regra presente em uma *chain* durante a ação do Netfilter. Cada pacote é analisado e verificado com as regras de uma *chain* e quando as características de um pacote coincidem com uma regra, a ação associada a essa regra é executada. Caso o pacote não coincida com nenhuma regra a

política padrão da *chain* é aplicada (AYUSO, 1999).

2.1.2 Iptables

Iptables foi criado por Rusty e é uma interface de comando que possibilita criar regras para as tabelas do Netfilter. Essas regras são responsáveis pela forma como o fluxo serão tratados, por exemplo: realizar operações como filtragem de pacote. Uma regra do Iptables é composta por parâmetros que determinam as características dos pacotes que estão em conformidade com a regra que está sendo estabelecida (AYUSO, 2006). Uma regra pode ser escrita seguindo a sintaxe exibida na figura 2.1;

Figura 2.1: Sintaxe de regra iptables.



Fonte: acervo pessoal.

2.1.2.1 Regra Iptables

Uma regra *Iptables* é composta basicamente por cinco partes como demonstrada na figura 2.1. No item A é definida em qual tabela a regra será inserida, no exemplo a regra será inserida na tabela FILTER. O item B define uma ação que se quer executar: " -A"para adicionar uma regra ao final da *chain*, " -I"para adicionar uma regra no inicio da *chain*, ou se seguido de um numero pode-se definir a posição em que a regra sera inserida na *chain*, " -D"para remover uma regra em uma *chain*. Após definir a ação, o item C define em qual *chain* a regra será adicionada, no exemplo a regra será adicionada na *chain* INPUT. O item D estará definindo características que o pacote deve possuir para ser tratado pela mesma, a quarta e ùltima parte define a ação a ser executada com o pacote que possuir as características da definidas na rega. A regra *Iptables* da figura 2.1 está definindo que a regra será inserida na tabela *filter*, na *chain INPUT*, filtrará pacotes TCP que se comunicam pela porta 22 que estão sendo recebido pelo dispositivo, pela ação definida na última parte da regra esses pacotes serão descartados.

2.2 SOCKET

Uma conexão de socket fornece um canal ponto a ponto de comunicação de via dupla, onde é possível que processos do sistema operacional possam trocar informações com outros processos que estejam em um mesmo host ou em hosts diferentes através de uma rede (BESAW, 1987). O Socket teve origem na Universidade de Berkeley, Califórnia, como sendo a API (Application Programming Interface) de desenvolvimento do protocolo TCP/IP para o ambiente UNIX (COMER,1993). A API de socket fornece um grupo de operações chamadas de primitivas, que facilitam o uso de funções fornecidas pela camada de transporte. Em (ANDREW S. TANENBAUM, 2010), no tópico 6.1.3 Berkeley Sockets, o socket e suas primitivas são abordados com maiores detalhes.

2.3 CRIPTOGRAFIA

Criptografia é o termo utilizado para técnicas e métodos de uma forma codificada ou cifrada. Segundo (ANDREW S. TANENBAUM, 2010), a escrita de uma mensagem de forma cifrada é conversão de caractere a caractere sem considerar a estrutura linguística da mensagem. Já a escrita de uma mensagem por código, consiste em substituir uma palavra inteira por um símbolo ou por outra palavra.

2.3.1 Criptografia Simétrica

Nesse modelo de criptografia os algoritmos de cifra utilizam um conjunto de caracteres referido como chave para codificar uma mensagem legível em um código ilegível, o algoritmo que irá descodificar essa mensagem utilizará a mesma chave, utilizada para cifrar a mensagem, para transformar o código ilegível em uma mensagem legível novamente. Dessa forma é necessário que os interlocutores envolvidos em uma comunicação cifrada possuam a chave para cifrar e decodificar as mensagens. Em caso das mensagens serem interceptadas, mesmo tendo conhecimento do algoritimo utilizado para codificar as mensagens, sem a chave correta não será possível ter acesso ao conteúdo legível (OLIVEIRA, 2012).

2.3.2 Criptografia Assimétrica

Esse modelo de criptografia foi proposto em 1976, na Universidade de Stanford, esse sistema, diferente do sistema simétrico, faz o uso de um par de chaves distintas, uma chave para o processo de criptografia e outra para descriptografar. Para que seja possível realizar uma comunicação utilizando essa técnica, é preciso que: primeiro se defina o algoritmo para criptografia e descriptografia e a suas chaves, o algoritmo e a chave utilizados para cifrar as mensagens devem ser públicos para que qualquer interlocutor da comunicação tenha acesso, e a chave e o algoritmo que são utilizados para descriptografar uma mensagem devem ser públicos, de forma que cada interlocutor possua sua chave privada para a descriptografar (ANDREW S. TANENBAUM, 2010).

2.4 TRABALHOS RELACIONADOS

A cerca da proposta dessa monografia existem duas soluções, uma delas é apresentada por (BRUNO COLACO LUIS PINTO, 2004), que propõe uma maneira de trabalhar com *firewalls* e outros serviços de forma remota. Para isso, um *middleware* executado em um servidor Web Apache, realiza a troca de mensagens com o cliente remoto através de um Web Browser. Nesse cenário é disponibilizado aos usuários, com acesso ao sistema, um meio para operar serviços remotamente. Porém, não é tratado de forma alguma o controle de acesso a recursos disponíveis a cada usuário, possibilitando que qualquer utilizador com acesso ao portal WEB aplicar modificações no *firewall*, que podem afetar de forma indesejada as redes que estão sobre o domínio desse *firewall*. A segunda solução (FRANCISCO JOSé CANDEIAS FIGUEIREDO, 2001) apresenta uma proposta de implementar o acesso remoto a *firewall* utilizando VPNs, o autor propõe o uso do software FreeS/WAN para criar o tunelamento entre um usuário e o dispositivo de firewall. Nessa solução é citado a complexidade da implementação dos *gateways* de VPN em uma rede, e a soluções não apresenta formas de controle de acesso ações que o usuário pode executar.

3 DESENVOLVIMENTO

Este capítulo abordará elementos pertinentes para o desenvolvimento da aplicação. Serão descritos tópicos referentes ao: funcionamento da aplicação, o protocolo de comunicação desenvolvido, ambiente para o controle de acesso e a implementação do servidor e o cliente da aplicação. Também são descritas as ferramentas utilizadas durante a implementação e testes realizados com a aplicação.

3.1 DESCRIÇÃO DO FUNCIONAMENTO DA APLICAÇÃO

Tendo em mente a conjuntura do ambiente de rede apresentado no Capítulo 1.2 e visando oferecer uma alternativa para as soluções propostas na seção trabalhos relacionados (2.4), foi desenvolvido uma aplicação denominada Chumbo. A estrutura da aplicação é composta por dois módulos principais, sendo um módulo servidor e um cliente. Além disso, a interação entre um usuário e o módulo cliente é através de uma CLI (Command-line interface). O estabelecimento da conexão entre o módulo cliente e o módulo servidor é realizada através da criação de um socket do tipo TCP/IP. Para implementar tal funcionalidade foi utilizado as funções oferecidas pelos módulos socket e ssl da linguagem Python 3.4. Em conjunto, esses dois módulos executam todo o procedimento para o estabelecimento de uma conexão entre dois hosts distintos. A figura 3.1 representa a arquitetura de funcionamento da aplicação Chumbo.

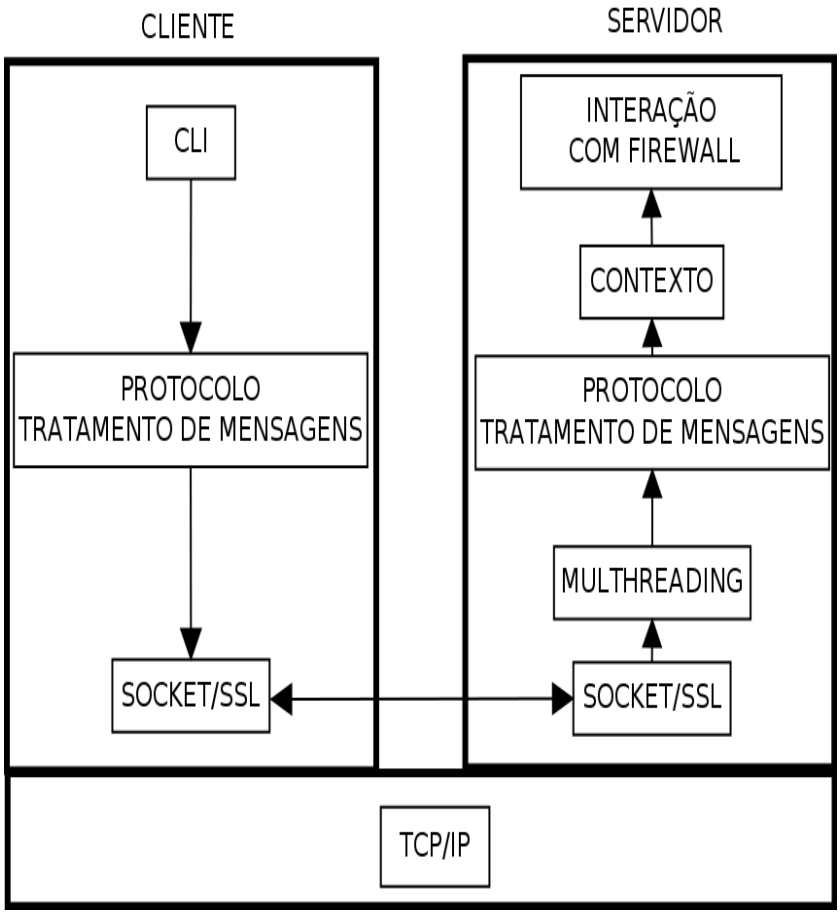
3.2 DEFINIÇÃO DO PROTOCOLO DAS MENSAGENS

Para realizar a comunicação referente as funções dos módulos foi desenvolvido um protocolo próprio e com vocabulário e sintaxes específicas para a aplicação. Nesse capítulo são tratados tópicos relevantes para a construção desse protocolo. A Figura 3.2, representa o fluxo das mensagens trocadas entre o módulo cliente e o módulo servidor.

3.2.1 Especificação do serviço

Esse protocolo tem como objetivo a troca de mensagens de texto entre dois interlocutores, através de um socket escrito em Python. O protocolo trabalha com a troca de mensagens de forma *half-duplex*, ou seja, a transmissão das mensagens é bidirecional entre o cliente e o servidor, porém a transmissão só ocorrerá em um sentido por vez.

Figura 3.1: Arquitetura da aplicação Chumbo.



Fonte: acervo pessoal.

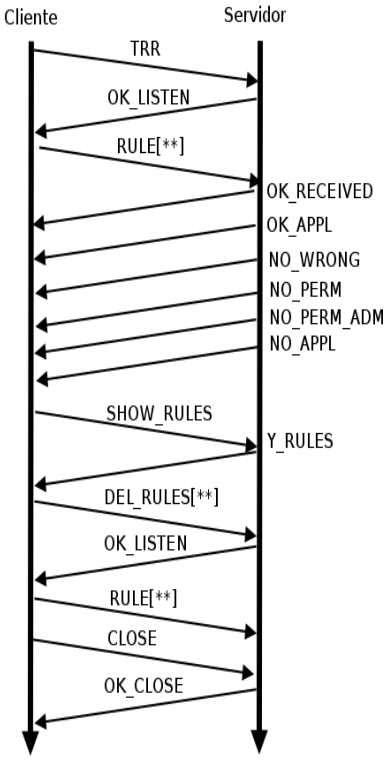
3.2.2 Restrições do ambiente

O cenário em que o protocolo estará operando é composto minimamente por um cliente e um servidor da aplicação Chumbo, e um socket para comunicação. O usuário utilizando a aplicação cliente, envia uma mensagem ao servidor, a mensagem é processada e o servidor retorna uma outra mensagem ao cliente. Ao longo da transmissão das mensagens, o meio de comunicação é suscetível a eventuais erros, como perda e duplicação, porém, esse tipo de evento é tratado pelo módulo Python que estabelece o socket de comunicação.

3.2.3 Vocabulário de mensagens

O protocolo utilizado na ferramenta possui um vocabulário próprio de mensagens na tabela 3.1 estão descritas respectivamente as mensagens e suas funções no protocolo.

Figura 3.2: Protocolo de mensagens para aplicação Chumbo.



Fonte: acervo pessoal.

3.2.4 Estrutura de uma regra de firewall

Para que seja possível o envio de uma regra pela aplicação Chumbo, é preciso que a regra de *firewall* a ser submetida, seja escrita seguindo uma sintaxe específica. Para ser aceita e interpretada pelo módulo servidor, a estrutura de uma regra deve ser elaborada de acordo com a sintaxe específica. A sintaxe para a composição de uma regra *firewall* consiste na inserção de marcações ao longo da regra de *firewall* que será submetida. Essas marcações tem como objetivo sinalizar para o *software* quais são os parâmetros de maior importância da regra que serão tratados, e dessa forma torna possível a verificação e manipulação desses parâmetros. Na Tabela 3.2, é descrito as marcações utilizadas reconhecidas pelo software e seus empregos.

3.3 DEFINIÇÃO DO AMBIENTE DE CONTROLE DE ACESSO

Um dos requisitos para o desenvolvimento da proposta da aplicação é oferecer uma forma controle de acesso e contexto das ações que o usuário poderá executar no *firewall* de borda. Para um cenário de implementação inicial para criar o ambiente de controle de acesso

Tabela 3.1: Vocabulário de mensagens do protocolo e suas funções

Vocabulário de mensagem	Ação
TRR	requisição de permissão, do cliente para início da transmissão de uma mensagem.
OK_LISTEN	mensagem de confirmação, referente a ação que antecedeu a mensagem.
RULE[**]	todo o conjunto de caracteres que estiverem entre os colchetes, é tratado como uma regra possível de ser aplicada no firewall.
OK_APPL	mensagem que sinaliza que uma regra foi aplicada com sucesso no firewall.
NO_APPL	mensagem que sinaliza que uma ação não foi executada com sucesso .
NO_WRONG	mensagem que demarca que a ação que antecede essa mensagem está escrita de uma forma fora da sintaxe conhecida pelo que o sistema.
NO_PERM	essa mensagem demarca que a regra que o cliente tentou aplicar no firewall, contém endereços IP que pertencem a um bloco de rede diferente do domínio do cliente que tentou aplicar a regra.
NO_PERM_ADM	essa mensagem demarca que a regra que o cliente tentou aplicar no firewall, contém endereços, elementos ou palavras chaves, cujo o administrador do firewall de borda não permite o uso.
SHOW_RULES	essa mensagem indica ao módulo servidor que o cliente está requisitando todas as regras que competem ao seu bloco de rede.
Y_RULES[**]	essa mensagem conterá todas as regras do cliente que a requisitou.
DEL_REULES[**]	essa mensagem conterá uma regra a qual o cliente deseja excluir do seu grupo de regras presente no firewall de borda.
CLOSE	mensagem que sinalizara o encerramento da conexão entre os módulos cliente e servidor
OK_CLOSE	mensagem que demarca o recebimento da requisição para que seja encerrada a conexão entre o módulo cliente e servidor.

Fonte: autoria própria.

para a ferramenta é utilizado uma base de dados estrutural.

3.3.1 Estrutura do banco de dados

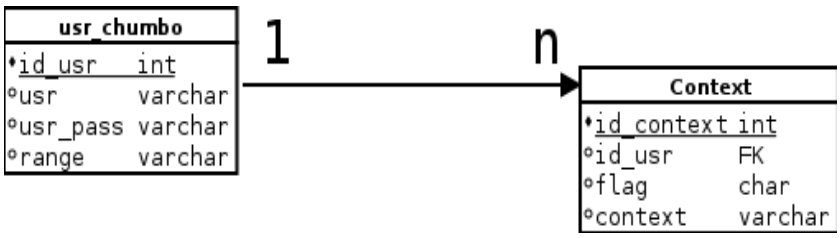
A base de dados do sistema é responsável por armazenar os dados para autenticação dos administradores de segmento. Os blocos de endereços IPs pertencentes a cada um desses administradores e as informações que irão compor o contexto de cada administrador. A figura 3.3 apresenta o diagrama ER da base de dados.

Tabela 3.2: Sintaxe de marcações para composição de regras de firewall

Marcação	Função
-s{**}	Marcar endereços IP de origem dos pacotes, aos quais será aplicado a regra.
–source{**}	Variação para marcar endereços IP de origem contido em uma regra.
-d{**}	Marcar endereços IP de destino dos pacotes, aos quais será aplicado a regra.
–destination{**}	Variação para marcar endereços IP de destino contido em uma regra.
–src-range{**}	Marcar um range de endereços IP para qual será aplicada uma regra.
mac{**}	Marcar o endereço mac contido em uma rega.
dport{**}	Marcar o número da porta de comunicação, contida na regra.
act{**}	Marca a ação que a regra irá executar sobre um determinado trafego.

Fonte: autoria própria .

Figura 3.3: Modelo ER da base de dados da ferramenta.



Fonte: acervo pessoal.

Na tabela "usr_chumbo"a coluna "usr_chumbo"será responsável por armazenar um identificador para cada usuário e a coluna "usr_pass"armazenará senhas dos usuários, utilizados para autenticação no sistema ao iniciar uma sessão. A coluna range armazenará o endereço da rede referente ao usuário cadastrado, o endereço é armazenado em formato CIDR.

A tabela "context"a coluna "flag"armazenará marcações que complementam os registros da coluna context, são duas possíveis marcações: NR, sinaliza para o sistema que o registro do campo context é uma regra específica registrada pelo administrador do *firewall* de borda, que não pode ser aplicada. A flag N indicia que registro do campo context é uma lista de palavras ou fragmentos de uma regra, onde regras submetidas ao sistema que contenham algum fragmento presente nessa lista, não será permitida sua execução no *firewall* de borda. A coluna context é armazenado informações que irão determinar o contexto que terá cada usuário.

3.4 FERRAMENTAS E LINGUAGENS DE PROGRAMAÇÃO

Para o desenvolvimento da aplicação proposta, a linguagem de programação utilizada foi Python, na sua versão 3.4. A opção de utilizar essa linguagem foi pelo fato da mesma ser uma linguagem de alto nível, possuir um vasto acervo de documentação, a facilidade oferecida por alguns módulos nativos da linguagem e por preferência do autor dessa monografia, tornando assim mais prático o processo de desenvolvimento. Como ambiente de desenvolvimento e testes foi utilizado um computador com configuração de: 8 gigabytes de memória RAM, processadores Intel® core™ i3-4010U de 1.70Ghz, executando Linux Debian 8.

3.4.1 Criando ambiente de controle para usuário

O ambiente de controle refere-se à delimitação das ações que cada usuário poderá realizar no *firewall*, bem como o tipo e o conteúdo das regras que serão submetidas pela aplicação Chumbo. O administrador principal do *firewall* irá cadastrar na base de dados da aplicação, regras que não devem ser aplicadas através da mesma, complementando o registro dessa regra na base de dados é preciso que seja indicado uma *flag* e um identificador de usuário. A *flag* irá sinalizar qual é o tipo daquele registro, se está restringindo a ação de uma regra completa ou se está restringindo a ação de regras que contenham um determinado fragmento na sua composição.

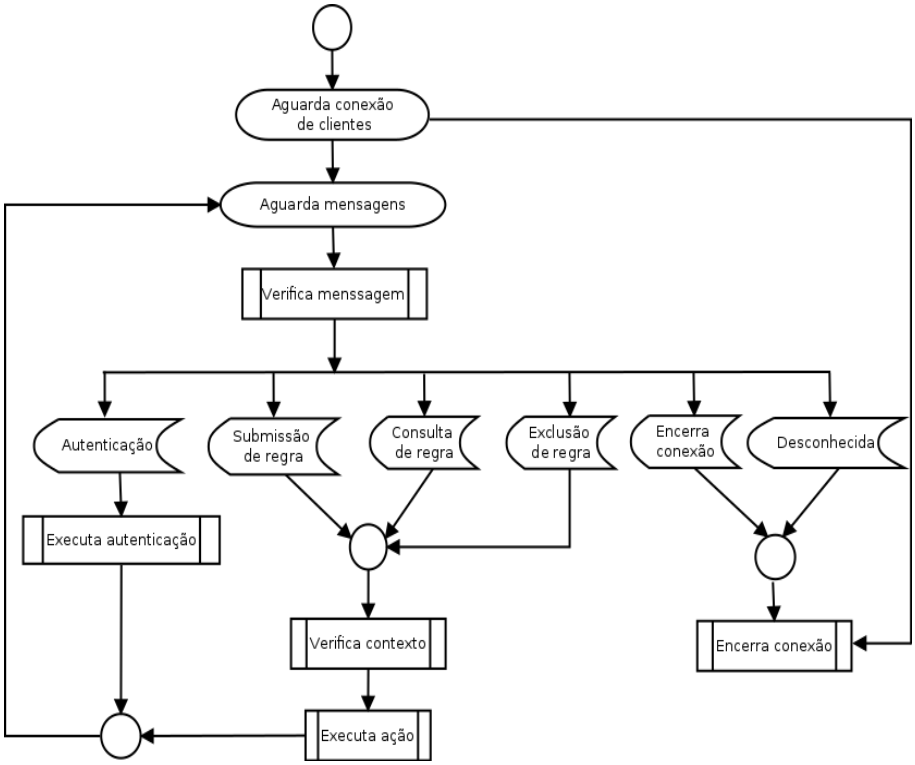
3.5 IMPLEMENTAÇÃO DO SERVIDOR

O módulo servidor da aplicação Chumbo, dividida em cinco submódulos, nesse capítulo serão abordados os o funcionamento desses módulos e suas funções, o diagrama de estados de transição representado na figura 3.4 demonstra o funcionamento do servidor da aplicação Chumbo.

3.5.1 Autenticação de Usuário

Após criar um nova conexão entre os módulos cliente e servidor, junto a isso é iniciado uma nova *thread* e atrelada ao socket recém criado. Essa *thread* irá tratar da comunicação entre os dispositivos. Nessa *thread* é executado um laço de repetição que monitora as mensagens oriundas do socket que esta atrelado a mesma, o laço de repetição permanecerá em execução até que a thread seja encerrada. A figura 3.5 demonstra o código usado para criar o socket.

Figura 3.4: Diagrama de estado de transição do servidor.



Fonte: acervo pessoal.

As três primeiras linhas indicam: o endereço de IP do host hospedeiro do servidor, a porta de que receberá as conexões dos clientes e o tamanho do *buffer* utilizado para o envio das mensagens. Em seguida utilizando a função "create_default_context", é estabelecido o contexto do socket ssl aonde é apontado o certificado digital e a chave desse certificado, que serão usados para cifrar as mensagens. Após isso, as próximas linhas definem: o protocolo de comunicação utilizado no socket aonde o parâmetro "SOCKET_STREAM"indica que o protocolo será TCP/IP e por ùltimo é indicado os endereços IP e porta para comunicação do servidor.

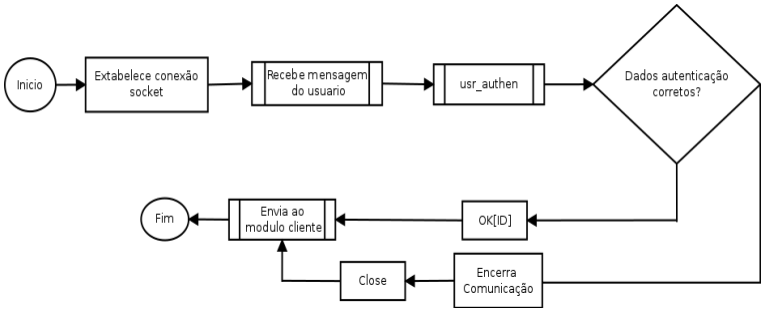
Figura 3.5: Código de socket na aplicação servidor.

```
TCP_IP = '0.0.0.0'
TCP_PORT = 3004
BUFFER_SIZE = 20000
#contexto SSL
context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
context.load_cert_chain(certfile="server.cert", keyfile="server.key")
#cria o socket
tcpServer = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpServer.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
tcpServer.bind((TCP_IP, TCP_PORT))
threads = []
```

Fonte: acervo pessoal.

O servidor ao receber uma mensagem que possui a marcação de uma mensagem de autenticação, em seguida essa mensagem é passada como parâmetro para a função "user_autent". A função "user_authen"tem o objetivo de extrair da mensagem os dados de login e senha para autenticação junto aos registros contidos na base de dados do módulo servidor. Caso os dados sejam válidos é retornado o id (número identificador) do usuário para a chamada principal da função "user_authet". O identificador retornado da função é encapsulado com uma flag "OK"e reenviado para o cliente que solicitou a autenticação. Caso os dados recebidos pelo servidor, quando solicitada a autenticação, não estiverem registrados na base de dados é retornado uma mensagem com uma *flag* "close", encerrando a conexão. A figura 3.6 demonstra o fluxo das informações para a autenticação de um usuário na aplicação Chumbo.

Figura 3.6: Fluxograma de autenticação de usuário no servidor.



Fonte: acervo pessoal.

3.5.2 Aplicação de regras

Ao receber uma mensagem com do tipo TRR, o servidor irá responder com uma mensagem de OK_LISTEN, para que o cliente possa dar sequência e submeter sua regra. Após o cliente submeter sua regra, a mensagem será analisada em busca das marcações estabelecidas pelo protocolo do sistema. Se não for encontrado a marcação "RULE"no início da mensagem, é retornado uma mensagem do tipo "NO_WRONG", indicando ao cliente que a regra está escrita fora do padrão aceito pelo módulo servidor da aplicação.

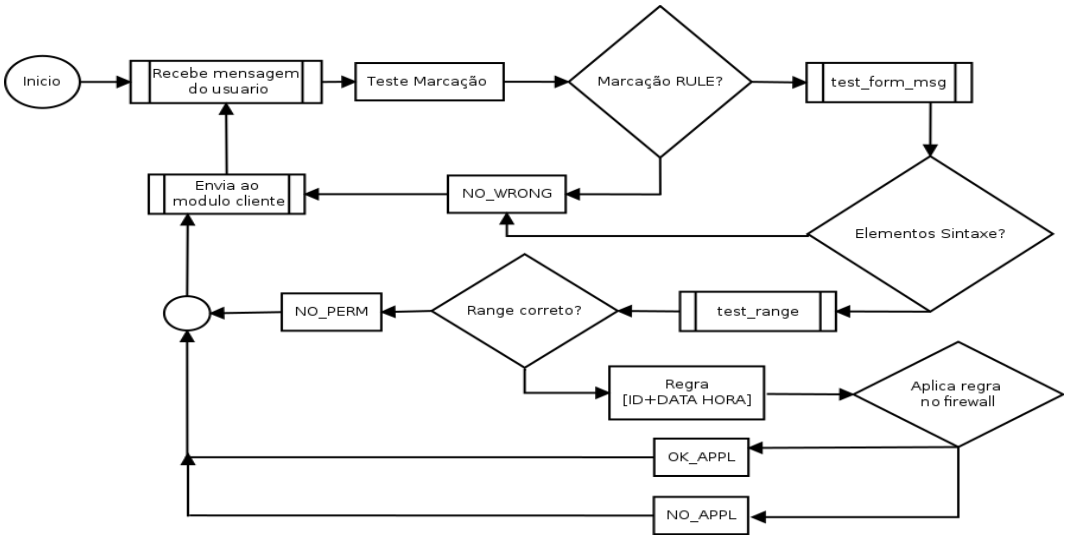
Caso a mensagem possua a marcação "RULE", a mensagem irá passar por uma série de funções que irão tratar e analisar o seu conteúdo. A primeira função é a "test_form_msg", que irá verificar a existência da *flag* "id"e a presença de algum outro elemento de sintaxe usada para compor uma regra de *firewall*. Caso a mensagem não contenha nenhum das flags testadas é retornado ao usuário uma mensagem do tipo "NO_WRONG"e ficará aguardando novas

mensagens. Passando pelo esse segundo teste é verificado ainda na função "test_form_msg", a existência de algum caractere especial, que possa gerar ambiguidade na interpretação da regra pelo *firewall*, como por exemplo, os caracteres de: cerquilha, arroba, cifrão, porcentagem, "E"comercial, contra barra e o sinal de igualdade. O processo para aplicar uma regra

Após passar pela verificação de sintaxe da mensagem, a próxima etapa é testar os endereços IP que essa regra irá afetar. Para isso são extraídos os endereços IP que estão contidos nas marcações que indicam a origem e/ou destino da comunicação e passada por parâmetro junto como id do cliente para a função "test_range". A função "test_range"irá consultar se os endereços contidos na regra submetida, estão dentro do range de endereços IPs pelo qual o usuário é responsável. Caso os endereços IPs contidos na regra estejam fora do range de endereço de responsabilidade do cliente, será retornado para o cliente uma mensagem do tipo "NO_PERM"e aguarda novas mensagens.

Se a regra não possuir impedimentos será acrescido a mesma uma marcação contendo o identificador do usuário que submeteu a regra junto com a data e hora da execução. Após esse tratamento a regra é aplicada ao firewall, caso a regra seja aplicada com sucesso, retorna uma mensagem ao usuário do tipo "OK_APPL", caso ocorra um erro ao aplicar a regra é retornado ao usuário uma mensagem do tipo "NO_APPL".

Figura 3.7: Fluxograma de autenticação de usuário no servidor.

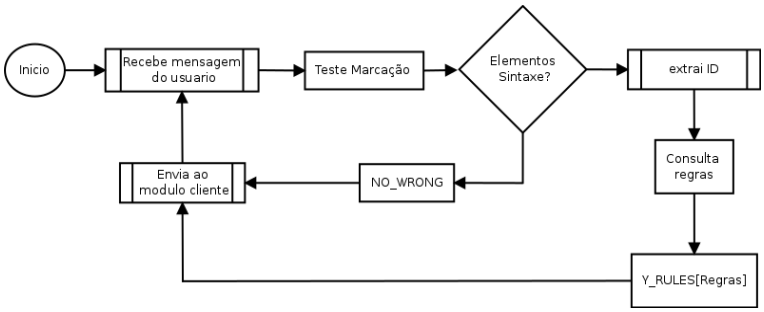


Fonte: acervo pessoal.

3.5.3 Consulta de Regras

Ao receber uma mensagem do tipo "SHOW_RULES", o módulo servidor irá verificar se a mensagem está escrita de acordo com a sintaxe definida pelo sistema e se existe a flag de identificação do usuário na mensagem. Se a mensagem não possuir identificação do usuário ou estiver fora da sintaxe estipulada é retornado para o módulo cliente uma mensagem do tipo "NO_WRONG". Se existir a identificação, o sistema irá extrair do arquivo que armazena as regras de *firewall*, todas as regras que possuem a identificação do usuário que solicitou. Após extrair essas regras o sistema irá encapsular todas em forma de uma lista de elementos e enviará ao cliente em forma de uma mensagem do tipo "Y_RULES". O fluxograma da figura 3.8 demonstra o funcionamento do processo de consulta de regras através da aplicação Chumbo.

Figura 3.8: Fluxograma de consulta de regras de usuário no servidor.

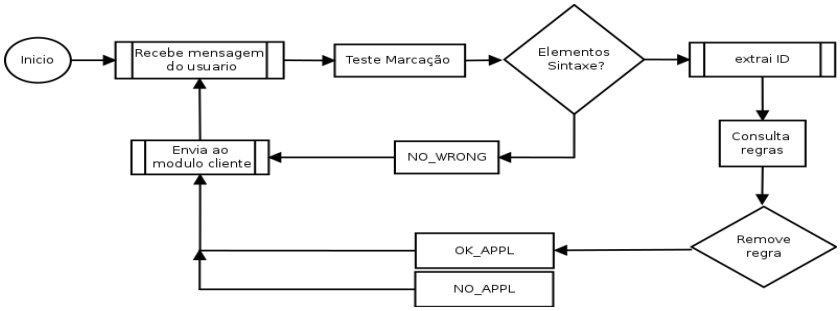


Fonte: acervo pessoal.

3.5.4 Exclusão de Regras

Ao receber uma mensagem do tipo "DEL_RULES", o módulo servidor irá verificar se a mensagem está escrita de acordo com a sintaxe definida pelo sistema e se existe a flag de identificação do usuário na mensagem. Se a mensagem não possuir identificação do usuário ou estiver fora da sintaxe estipulada é retornado para o módulo cliente uma mensagem do tipo "NO_WRONG". Se existir a identificação, o sistema irá executar o comando para eliminar a regra submetida e retornará uma mensagem do tipo "OK_APPL"ao usuário. O fluxograma da figura 3.9 demonstra o funcionamento do processo para autenticação de um usuário.

Figura 3.9: Fluxograma de exclusão de regras de usuário no servidor.

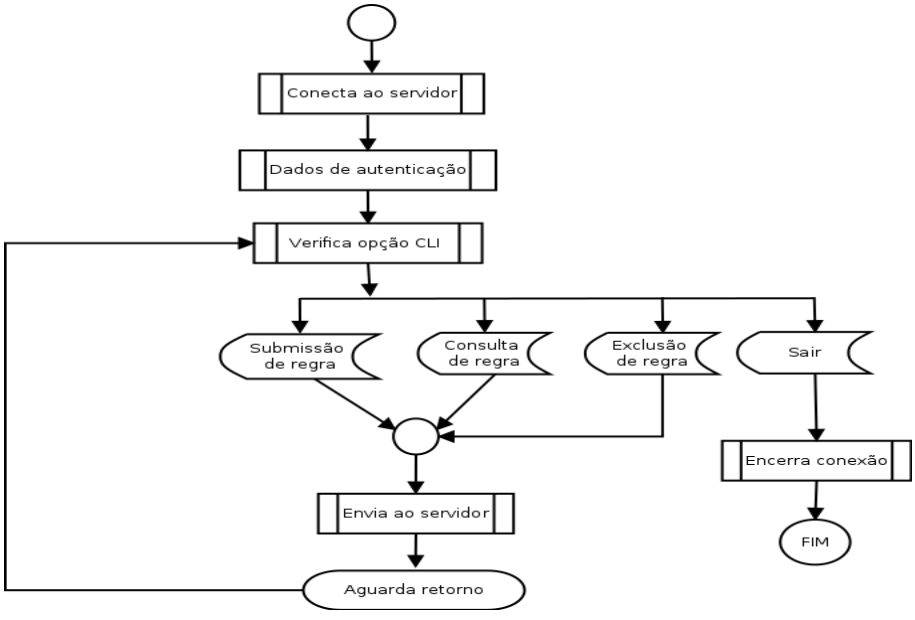


Fonte: acervo pessoal.

3.6 IMPLEMENTAÇÃO DO CLIENTE

Nesta seção será abordado o funcionamento dos módulos principais que compõem a aplicação cliente ,bem como sua interação com o módulo servidor. Serão tratados tópicos referentes aos processos de: autenticação de usuário, implementação da CLI, submissão, consulta e exclusão de uma regra. Para demonstrar o funcionamento da aplicação cliente, a figura

Figura 3.10: Diagrama de estado de transição do cliente.



Fonte: acervo pessoal.

3.6.1 Socket e autenticação

Para estabelecer a conexão com o servidor da aplicação Chumbo foi implementado o um socket, o parte do código desse socket está representado na 3.11. As quatro primeiras linhas do código são responsáveis por definir os parâmetros de identificação de host, porta de comunicação, o tamanho do buffer para o socket. As próximas linhas definem o contexto do socket ssl, é definida a versão do protocolo SSL a ser usada, parâmetro requerendo que o socket use o certificado digital e a localização desse certificado. Por fim é definindo o protocolo a ser usado, nome do servidor e inicializada a conexão.

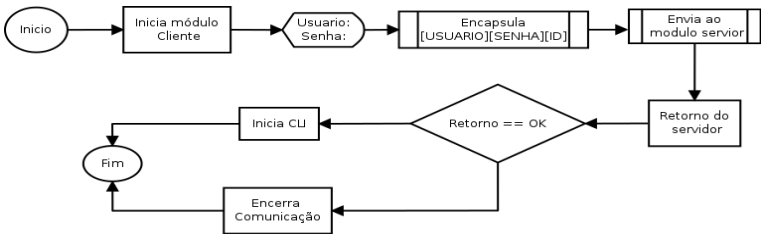
Figura 3.11: Código de socket na aplicação cliente.

```
host = socket.gethostname()
port = 3004
BUFFER_SIZE = 2000000
#contexto SSL
context = ssl.create_default_context()
context = ssl.SSLContext(ssl.PROTOCOL_SSLv23)
context.verify_mode = ssl.CERT_REQUIRED
context.check_hostname = True
context.load_verify_locations("server.cert")
#socket seguro
ClienteA = context.wrap_socket(socket.socket(socket.AF_INET,socket.SOCK_STREAM),
server_hostname="chumbo",do_handshake_on_connect=True)
#conecta com o servidor
ClienteA.connect_ex((host, port))
```

Fonte: acervo pessoal.

Ao executar o módulo cliente é solicitado que o usuário submeta dados de login e senha para autenticação no sistema. Após digitar esses dados, o módulo cliente irá encapsular esses dados e adicionar uma marcação para indicar qual elemento da mensagem se refere ao usuário e qual se refere a senha.

Figura 3.12: Fluxograma de autenticação de usuário no sistema.



Fonte: acervo pessoal.

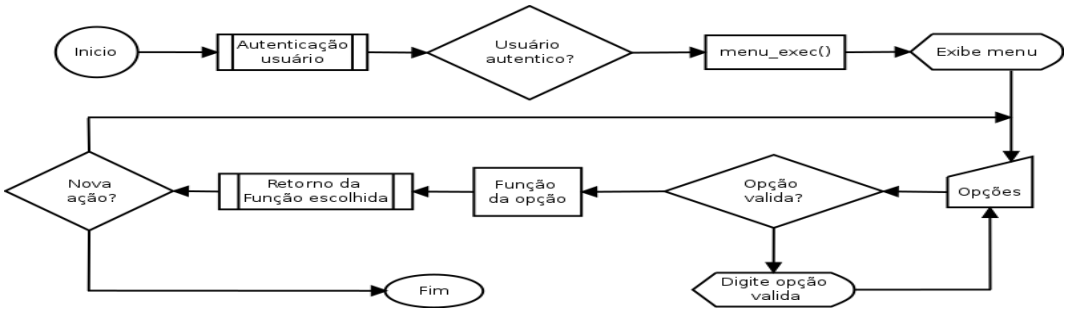
Os dados encapsulados são enviados ao módulo servidor, onde o mesmo irá analisar os dados se realizar a autenticação. Caso a autenticação seja realizada com sucesso, ao receber

a mensagem de confirmação enviada pelo módulo servidor, o módulo cliente inicia a função para execução do CLI. O fluxograma da figura 3.12 demonstra o funcionamento do processo de autenticação.

3.6.2 Implementação de CLI

Command Line é uma interface para interação entre o usuário e um sistema computacional. A interação ocorre através de comandos específicos inseridos pelo usuário através de um console ou terminal. Os comandos são interpretados pelo sistema computacional, retornando uma resposta específica. O módulo cliente da ferramenta Chumbo possui uma CLI para auxiliar o usuário a executar suas funções. A CLI oferece duas funções para navegação entre as opções da aplicação. O fluxograma da figura 3.13 demonstra o funcionamento da interação entre o usuário e a interface CLI da aplicação. Após a autenticação é exibido ao usuário a CLI para

Figura 3.13: Fluxograma de interação de usuário e CLI.



Fonte: acervo pessoal.

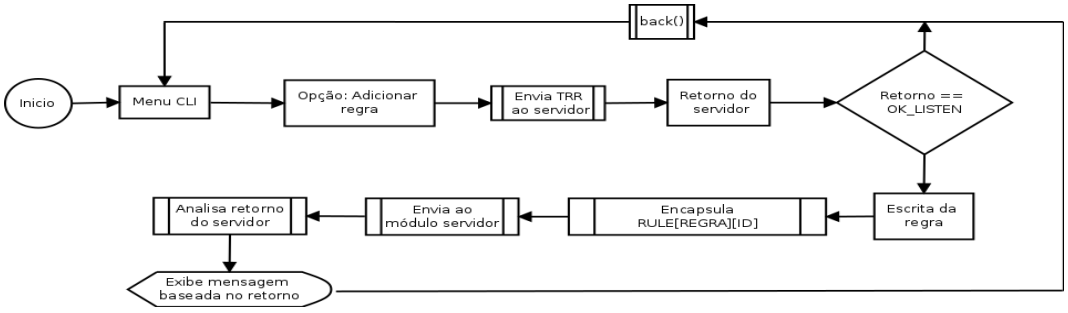
realizar a interação, a primeira tela exibe um menu com as opções para adicionar uma regra, mostrar as regras e deletar uma regra. O usuário seleciona uma das opções digitando o número referente a uma das funções a ser executada. Se o número digitado não for um número válido para as opções apresentadas é exibido ao usuário uma mensagem informando para ser digitado um número válido.

3.6.3 Aplicação de regras

A primeira opção da aplicação é para a adição de regras ao *firewall* é executada a função "add_rules", essa função envia uma mensagem do tipo "TRR" ao módulo servidor e aguarda receber uma resposta contendo uma mensagem do tipo "OK_LISTEN". Caso a resposta rece-

bida seja "NO_WRONG"é executado a função “back”, que exibirá uma mensagem informando o erro ao usuário e retornará a CLI do módulo cliente para o menu principal da aplicação. A regra que será ser submetida deve ser escrita seguindo a sintaxe para marcação dos elementos (endereço de origem, destino) descrita pelo protocolo do sistema. Antes de enviar a mensagem o módulo cliente encapsula a mensagem, adicionando as *flags* de marcação “RULE[“ no início e “]” no final da mensagem. Após esse tratamento a mensagem é enviada ao módulo servidor. Posteriormente, o módulo cliente monitora a mensagem de retorno. As possíveis mensagens de retorno são: “OK_APPL”, “NO_APPL”, “NO_WRONG”, “NO_PERM”, “NO_PERM_ADM”, o significado dessas mensagens estão descritos na tabela 3.1. O fluxograma da figura 3.14 demonstra o funcionamento para a aplicação de uma regra ao firewall.

Figura 3.14: Fluxograma para submissão de uma regra ao firewall.

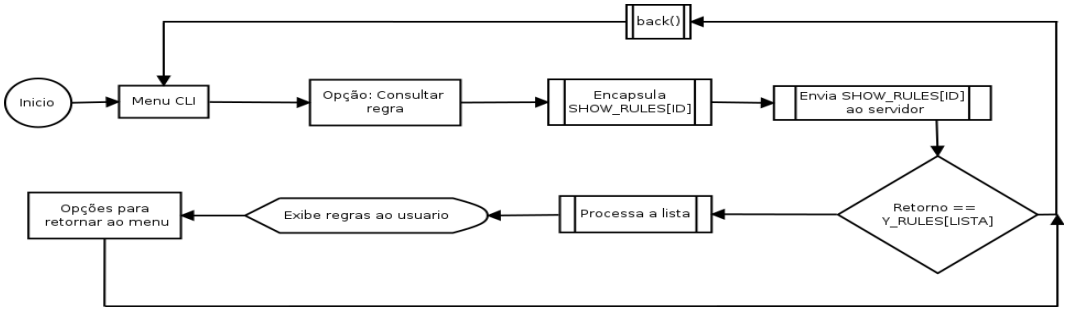


Fonte: acervo pessoal.

3.6.4 Consulta de regras

Ao selecionar uma a opção para mostrar suas regras, o cliente envia ao módulo servidor uma mensagem do tipo “SHOW_RULES” junto com o identificador do usuário, e aguarda uma mensagem do tipo “Y_RULES”. A mensagem Y_RULE, é uma lista de elementos, onde cada elemento é uma regra de *firewall* que possui identificação do usuário que solicitou. O modulo cliente extraí os elementos dessa lista e apresenta ao usuário. O fluxograma da figura 3.15 demonstra o funcionamento de uma consulta as regras do usuário ao firewall.

Figura 3.15: Fluxograma para consulta de regra ao firewall.

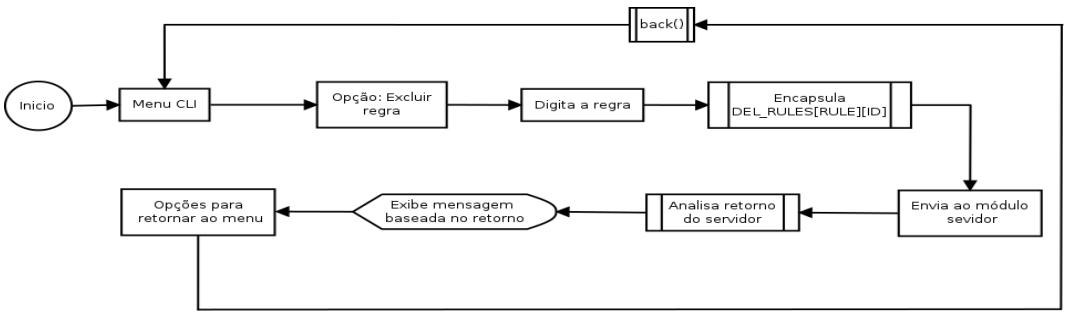


Fonte: acervo pessoal.

3.6.5 Exclusão de regras

A última função do módulo cliente é para a exclusão de regras, ao selecionar essa opção o usuário deve escrever a regra que deverá ser removida do *firewall*, seguindo a sintaxe definida pelo protocolo. A regra será encapsulada junto ao identificador do usuário e enviada ao módulo servidor. Após a mensagem para exclusão enviada , o módulo cliente irá monitorar as mensagens de retorno “OK_APP”, “NO_PPL”, “NO_WRONG”, “NO_PERM”, “NO_PERM_ADM”. O fluxograma da figura 3.16 demonstra o funcionamento para exclusão de uma regra do usuário no *firewall* de borda.

Figura 3.16: Fluxograma para exclusão de regra ao firewall.



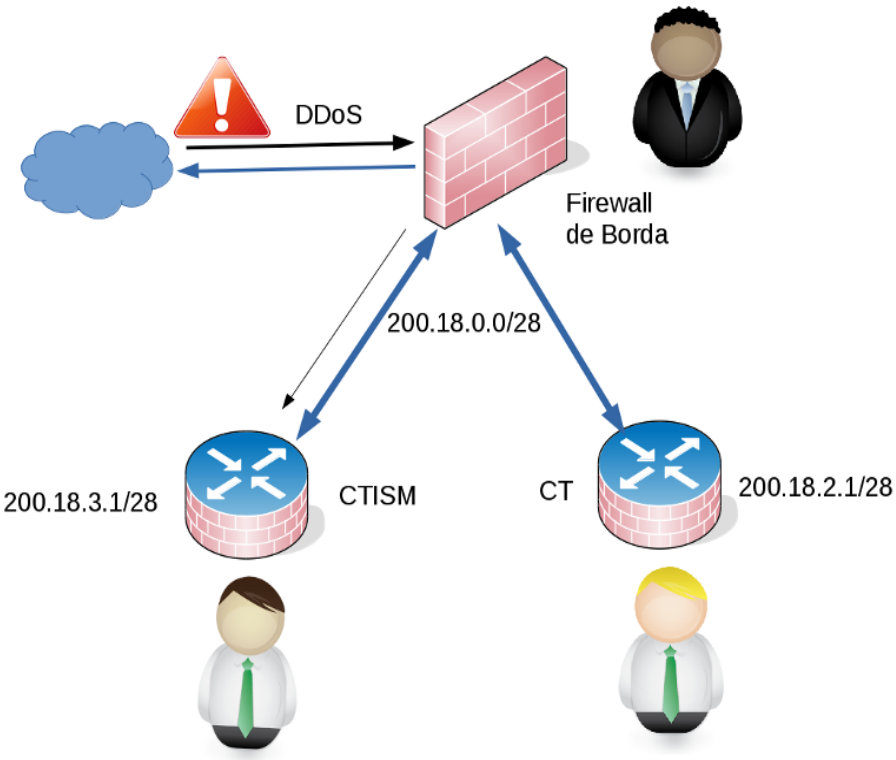
Fonte: acervo pessoal.

Em sequência a exposição dos métodos e fundamentos utilizados para a implementação do servidor e cliente da aplicação Chumbo, será realizado o processo de teste da aplicação e analisado os resultados obtidos.

4 TESTES E RESULTADOS

Nesse capítulo serão apresentados os testes realizados e os resultados obtidos. Para a execução dos testes da ferramenta Chumbo foi implementado um cenário, utilizando o software Virtualbox, para emular os elementos do cenário de testes. Os hosts utilizados no cenário são máquinas virtuais executando Linux Debian 8, com 800 Megabytes de memória RAM e 10 Gigabytes de armazenamento. A figura 4.1 representa o cenário elaborado. Esse cenário exem-

Figura 4.1: Topologia de cenário para testes

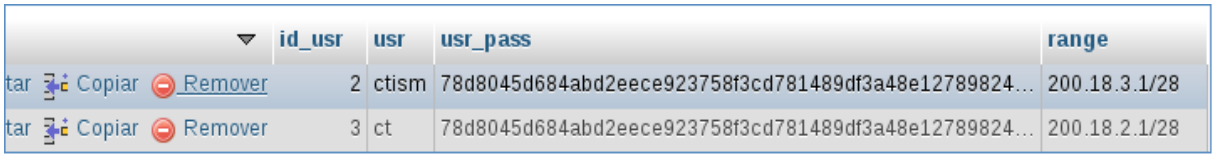


Fonte: acervo pessoal.

plifica uma rede composta por duas sub-redes protegidas cada uma por um *firewall* e ambos blocos conectados ao *firewall* de borda. O *firewall* de borda está conectado com os dois blocos de redes, através da rede 200.18.0.0/28 e com uma conexão para uma rede 172.23.0.0/28, que exemplifica uma conexão com a internet para o ambiente de teste, o *firewall* de borda está executando: o software Iptables/Netfilter para realizar o filtro de pacotes, o software Wireshark para

análise do tráfego de rede e um módulo servidor da aplicação Chumbo. O bloco CTISM possui o bloco de sub-rede 200.18.3.1/ 28 e o bloco CT possui a sub-rede de endereço 200.18.2.1/28. Na rede 172.23.0.0 ainda também está conectado um host, que irá realizar a função de atacante. Com o objetivo de simular um ataque real à rede implementada foi desenvolvido uma pequena aplicação em Python para realizar essa função. Essa aplicação recebe como parâmetro endereços IP e inicializa um laço de repetição enviando 41 pacotes do tipo ICMP simultâneos para cada endereço IP setado como parâmetro simulando um ataque do tipo ICMP Flood. Para o cenário de testes da aplicação Chumbo foi criado um conjunto de usuário/senha e um contexto para cada um dos blocos. As figuras 4.2 e 4.3 demonstram, respectivamente, a estrutura das tabelas de usuário e contexto da base de dados.

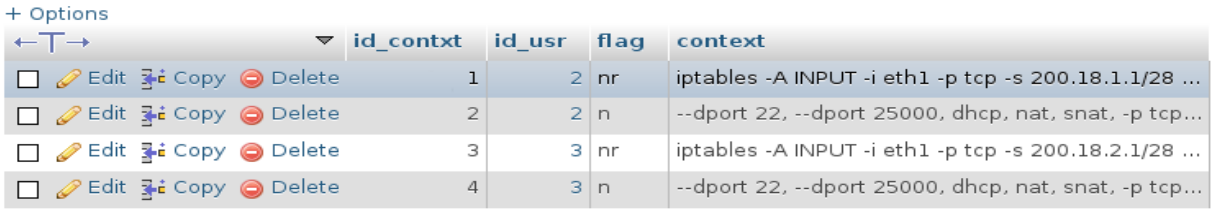
Figura 4.2: Tabela contendo dados de autenticação de usuários.



	id_usr	usr	usr_pass	range
tar Copiar Remover	2	ctism	78d8045d684abd2eece923758f3cd781489df3a48e12789824...	200.18.3.1/28
tar Copiar Remover	3	ct	78d8045d684abd2eece923758f3cd781489df3a48e12789824...	200.18.2.1/28

Fonte: acervo pessoal.

Figura 4.3: Tabela de contexto de usuários



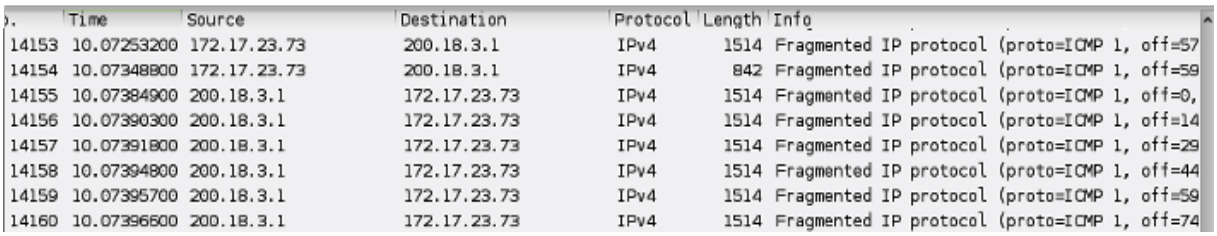
+ Options

	id_contxt	id_usr	flag	context
Edit Copy Delete	1	2	nr	iptables -A INPUT -i eth1 -p tcp -s 200.18.1.1/28 ...
Edit Copy Delete	2	2	n	--dport 22, --dport 25000, dhcp, nat, snat, -p tcp...
Edit Copy Delete	3	3	nr	iptables -A INPUT -i eth1 -p tcp -s 200.18.2.1/28 ...
Edit Copy Delete	4	3	n	--dport 22, --dport 25000, dhcp, nat, snat, -p tcp...

Fonte: acervo pessoal.

Após montar o cenário foi executado a aplicação que gerou um fluxo de pacotes ICMP com destino os endereços IP 200.18.3.1 referentes ao bloco CTISM e ao endereço 200.18.2.1 referente ao bloco CT. Utilizando o software Wireshark para analisar o tráfego da rede no *firewall* foi possível visualizar os pacotes ICMP com origem aos endereços de rede do bloco CTISM e CT. As figuras 4.4 e 4.5 demonstram a captura dos pacotes que possuem como destino aos endereços 200.18.3.1 e 200.18.2.1, respectivamente. Como o módulo servidor da

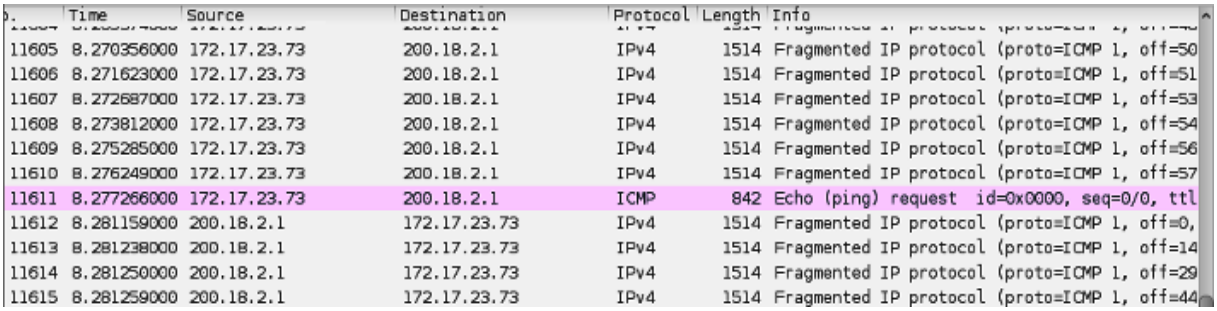
Figura 4.4: Captura de pacotes ICMP com destino ao endereço 200.18.3.1.



No.	Time	Source	Destination	Protocol	Length	Info
14153	10.07253200	172.17.23.73	200.18.3.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=57
14154	10.07348800	172.17.23.73	200.18.3.1	IPv4	842	Fragmented IP protocol (proto=ICMP 1, off=59
14155	10.07384900	200.18.3.1	172.17.23.73	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0,
14156	10.07390300	200.18.3.1	172.17.23.73	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=14
14157	10.07391800	200.18.3.1	172.17.23.73	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=29
14158	10.07394800	200.18.3.1	172.17.23.73	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=44
14159	10.07395700	200.18.3.1	172.17.23.73	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=59
14160	10.07396600	200.18.3.1	172.17.23.73	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=74

Fonte: acervo pessoal.

Figura 4.5: Captura de pacotes ICMP com destino ao endereço 200.18.2.1.



No.	Time	Source	Destination	Protocol	Length	Info
11605	8.270356000	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=50
11606	8.271623000	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=51
11607	8.272687000	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=53
11608	8.273812000	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=54
11609	8.275285000	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=56
11610	8.276249000	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=57
11611	8.277266000	172.17.23.73	200.18.2.1	ICMP	842	Echo (ping) request id=0x0000, seq=0/0, ttl
11612	8.281159000	200.18.2.1	172.17.23.73	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0,
11613	8.281238000	200.18.2.1	172.17.23.73	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=14
11614	8.281250000	200.18.2.1	172.17.23.73	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=29
11615	8.281259000	200.18.2.1	172.17.23.73	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=44

Fonte: acervo pessoal.

aplicação Chumbo em execução no dispositivo de *firewall*, foi inicializado dois módulos cliente simultâneos e realizada a autenticação de ambos na aplicação. A figura 4.6 e a figura demonstra o estabelecimento da conexão e autenticação dos clientes ao módulo servidor. No terminal

Figura 4.6: Tela de autenticação de dois usuários simultâneos.

```
root@debian:/home/toor# python3 clienteA.py
Para iniciar digite o usuario e senha!

login: ct
password: _
```

```
root@debian:/home/toor# python3 clienteB.py
Para iniciar digite o usuario e senha!

login: ctism
password: _
```

Fonte: acervo pessoal.

do servidor é possível visualizar o momento em que é estabelecida uma conexão do socket e o endereço que realizou a conexão. Após a autenticação dos módulos é exibido a interface CLI para interação do usuário. A figura 4.8 representa a interface do módulo cliente autenticada com o usuário CTISM. Em um segundo momento iniciou-se os testes para a submissão de re-

Figura 4.7: Tela de conexão de dois usuários simultâneos no servidor.

```
[+]Aguardando clientes...
[+] Nova conexao iniciada por 200.18.2.1:51804
[+]Aguardando clientes...
[+] Nova conexao iniciada por 200.18.3.1:39581
[+]Aguardando clientes...
```

Fonte: acervo pessoal.

Figura 4.8: Tela de interface CLI.

```
============================================================
Menu Principal
============================================================

)              *                )
(      ( /(            (  `        (     ( /(
)\     )\())      (    )\))(     ( )\    )\())
(((_)   ((_)\       )\   ((_)()\    )((_)  ((_)\
)\___    _((_)   _ ((_)  (_()((_) ((_)_     ((_)
((/ __|  | || |  | | |  | |  \/  |   | _ )   / _ \
 | (__   | __ |  | |_|  | |  |\/| |   | _ \  | (_) |
  \___|  |_||_|   \___/   |_|  |_|   |___/   \___/  V1.0


============================================================
Id do usuario: 2
1. Adicionar Regra
2. Mostrar Regra
3. Deletar Regra
0. Sair
Opcao:
```

Fonte: acervo pessoal.

gras ao *firewall*. A primeira tentativa de submissão de uma regra foi enviando ao sistema uma regra elaborada fora da sintaxe descrita pelo sistema. A figura 4.9 demonstra o resultado obtido. Após a tentativa do envio de uma regra escrita fora da sintaxe da aplicação foi realizada a tentativa de envio de uma regra elaborada seguindo a sintaxe exigida pela aplicação, porém com um endereço IP que está fora do contexto do usuário CTISM. A figura 4.10 representa a ação de controle de acesso fornecida pela aplicação. Outro teste realizado foi a submissão de uma regra contendo elementos, que foram estipulados como proibidos pelo administrador de borda, no contexto do usuário. A figura 4.11 demonstra o resultado da ação. O último teste realizado para o envio de regras foi a submissão de uma regra elaborada seguindo a sintaxe correta definida pela aplicação. O objetivo dessa regra é barrar o fluxo de pacotes ICMP disparado pela aplicação atacante. A figura 4.12 demonstra a submissão bem sucedida da regra ao *firewall*. Em sequência a submissão e execução da regra no *firewall* corretamente, utilizando o Wireshark para analisar o fluxo de pacotes ICMP na rede do *firewall* de borda, pode-se observar como representado na figura, que o tráfego de pacotes ICMP para rede 200.18.3.1 foi interrompido, porém o tráfego ICMP continua ativo para a rede 200.18.2.1, como pode ser observado na figura

Figura 4.9: Tela de submissão de uma regra elaborada errada.

```
Id do usuario: 2
================================================================================
Menu Principal
================================================================================

)             *                )
(     ( /(           (  `       (    ( /(
)\    )\())     (    )\))(    ( )\   )\())
(((_)  ((_)\      )\  ((_)()\   )((_) ((_)\
)\___   _((_)  _ ((_) (_()((_) ((_)_    ((_)
((/ __| | || | | | | |  \/  |  | _ )  / _ \
 | (__  | __ | | |_| | | |\/| |  | _ \ | (_) |
  \___| |_||_|  \___/  |_|  |_|  |___/  \___/  V1.0


================================================================================
Area para adicionar nova regras ao firewall!
4. Voltar
0. Sair
Digite a regra, seguindo a sintaxe do firewall destinatario, deixando todos os enderecos
ip envolvidos entre chaves
-> iptables -A FORWARD -i eth0 -O eth1 --destination 200.18.3.1 -p icmp --icmp-type
echo-request -j DROP
Regra escrita de forma errada, tente novamente!,(Pressione qualquer tecla)_
```

Fonte: acervo pessoal.

Figura 4.10: Tela de submissão de uma regra fora do contexto do usuário.

```
Id do usuario: 2
================================================================================
Menu Principal
================================================================================

)             *                )
(     ( /(           (  `       (    ( /(
)\    )\())     (    )\))(    ( )\   )\())
(((_)  ((_)\      )\  ((_)()\   )((_) ((_)\
)\___   _((_)  _ ((_) (_()((_) ((_)_    ((_)
((/ __| | || | | | | |  \/  |  | _ )  / _ \
 | (__  | __ | | |_| | | |\/| |  | _ \ | (_) |
  \___| |_||_|  \___/  |_|  |_|  |___/  \___/  V1.0


================================================================================
Area para adicionar nova regras ao firewall!
4. Voltar
0. Sair
Digite a regra, seguindo a sintaxe do firewall destinatario, deixando todos os enderecos
ip envolvidos entre chaves
-> iptables -A FORWARD -i eth0 -O eth1 --destination{200.18.2.1} -p icmp --icmp-type
echo-request act{-j DROP}
Regra nao pode ser aplicada, os enderecos da regra estao fora do seu bloco de ips!,
(Pressione qualquer tecla)_
```

Fonte: acervo pessoal.

4.13.

Figura 4.11: Tela de submissão de uma regra com elemento proibido no contexto do usuário

```
Id do usuario: 2
================================================================================
Menu Principal
================================================================================

)              *                )
(     ( /(            (  `        (    ( /(
)\    )\())     (    )\))(    ( )\   )\())
(((_)  ((_)\      )\  ((_)()\   )((_) ((_)\
)\___   _((_)  _ ((_) (_()((_) ((_)_    ((_)
((/ __| | || | | | | | |  \/  |  | _ )  / _ \
 | (__  | __ | | |_| | | |\/| |  | _ \ | (_) |
  \___| |_||_|  \___/  |_|  |_|  |___/  \___/  V1.0


================================================================================
Area para adicionar nova regras ao firewall!
4. Voltar
0. Sair
Digite a regra, seguindo a sintaxe do firewall destinatario, deixando todos os enderecos
ip envolvidos entre chaves
-> iptables -A FORWARD -i eth0 -O eth1 --destination{200.18.2.1} -dport{22}  act{-j DROP}
Regra nao pode ser aplicada, regra nao permitida pelo administrador de borda!,
(Pressione qualquer tecla)_
```

Fonte: acervo pessoal.

Figura 4.12: Tela de submissão de uma regra bem sucedida.

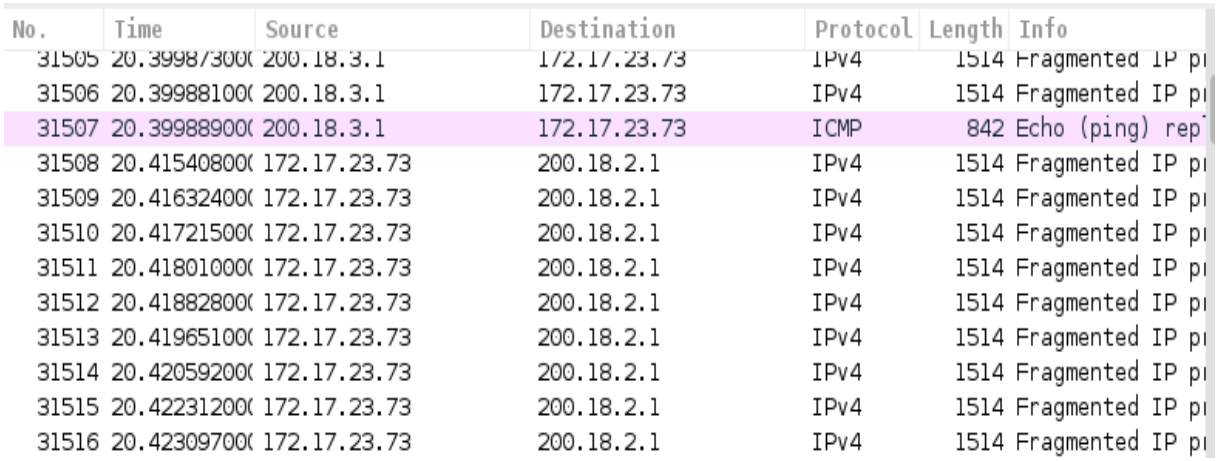
```
Id do usuario: 2
================================================================================
Menu Principal
================================================================================

)              *                )
(     ( /(            (  `        (    ( /(
)\    )\())     (    )\))(    ( )\   )\())
(((_)  ((_)\      )\  ((_)()\   )((_) ((_)\
)\___   _((_)  _ ((_) (_()((_) ((_)_    ((_)
((/ __| | || | | | | | |  \/  |  | _ )  / _ \
 | (__  | __ | | |_| | | |\/| |  | _ \ | (_) |
  \___| |_||_|  \___/  |_|  |_|  |___/  \___/  V1.0


================================================================================
Area para adicionar nova regras ao firewall!
4. Voltar
0. Sair
Digite a regra, seguindo a sintaxe do firewall destinatario, deixando todos os enderecos
ip envolvidos entre chaves
-> iptables -A FORWARD -i eth0 -O eth1 --destination{200.18.3.1} -p icmp  act{-j DROP}
Regra aplicada com sucesso!, (Pressione qualquer tecla)_
```

Fonte: acervo pessoal.

Figura 4.13: Tela de captura de pacotes ICMP após submissão de regra.



No.	Time	Source	Destination	Protocol	Length	Info
31505	20.3998730(	200.18.3.1	172.17.23.73	IPv4	1514	Fragmented IP pı
31506	20.39988100(	200.18.3.1	172.17.23.73	IPv4	1514	Fragmented IP pı
31507	20.39988900(	200.18.3.1	172.17.23.73	ICMP	842	Echo (ping) rep
31508	20.41540800(	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP pı
31509	20.41632400(	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP pı
31510	20.41721500(	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP pı
31511	20.41801000(	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP pı
31512	20.41882800(	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP pı
31513	20.41965100(	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP pı
31514	20.42059200(	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP pı
31515	20.42231200(	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP pı
31516	20.42309700(	172.17.23.73	200.18.2.1	IPv4	1514	Fragmented IP pı

Fonte: acervo pessoal.

5 CONSIDERAÇÕES FINAIS

A segurança em um âmbito empresarial é algo que impacta diretamente no rendimento de negócios, a capacidade e agilidade durante a realização dos processos executados por uma empresa e/ou entidade refletem diretamente no poder competitivo da mesma. A ausência de segurança pode afetar a capacidade e agilidade de operação de uma organização e isso gerará prejuízos(EMILIO TISSATO NAKAMURA, 2008).

Tendo em mente que em muitos casos a administração do *firewall* responsável pela segurança e confiabilidade dos dados trafegados na rede de uma organização, é centralizado em uma única equipe de administradores e dependendo da dimensão dessa organização, torna-se complexo para essa equipe ter conhecimento de forma ágil do cenário corrente de cada polo dessa instituição.

A principal contribuição dessa monografia é demonstrar os passos para realizar a implementação de uma ferramenta que torna possível a integração e controle de acesso de usuários a um *firewall*. Dessa forma descentralizando de uma forma contida, o poder de tomada de decisão das políticas de segurança em uma rede, tornando possível essa diminuir o tempo de resposta a um incidente de segurança, como por exemplo: através da ferramenta Chumbo um administrador de segmento pode submeter uma regra ao *firewall* para mitigar a ação de um tráfego malicioso.

Como demonstrado nos testes é possível observar que a solução apresentada nessa monografia, executa de forma satisfatória os objetivos propostos. Como intenção para trabalhos futuros estão: Efetuar a melhoria no socket de comunicação, e o sistema de criptografia das mensagens utilizando SSL/TLS, visando aumentar a confiabilidade da aplicação; efetuar a melhoria na sintaxe da ferramenta tornando-a mais precisa e; a implementação de um sistema de controle de acesso mais eficaz, e adição de módulos que possibilitem que um administrador local possua atrelado a ele múltiplas sub-redes diferentes.

REFERÊNCIAS

AKAMAI. **Q2 2016 State of the Internet Security Report | Akamai**. Acessado em Novembro/2016, https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/akamai-q2-2016-state-of-the-internet-security-report.pdf.

ANDREW S. TANENBAUM, D. J. W. **Computer Network**. 5th.ed. USA: Pearson, 2010.

AYUSO, P. N. **Netfilter's Connection Tracking System**. Acessado em Novembro/2016, https://www.netfilter.org/documentation/index.html.

AYUSO, P. N. **Netfilter's Connection Tracking System**. Acessado em Novembro/2016, https://people.netfilter.org/pablo/docs/login.pdf.

BESAW, L. Berkeley UNIX† System Calls and Interprocess Communication. **CS 640 Fall 1994 BSD Socket Reference**, [S.l.], 1987.

BRUNO COLACO LUIS PINTO, P. S. E. M. Mecanismos de Gestao Integrada para Firewall Appliances. **7a Conferencia sobre Redes de Computadores**, [S.l.], October 2004.

ELIZABETH D. ZWICKY, S. C. . D. B. C. **Building Internet Firewalls**. 2th.ed. USA: NOVATEC Co., Inc., 2000.

EMILIO TISSATO NAKAMURA, P. L. d. G. **Segurança de Redes em Ambientes Cooperativos**. 7th.ed. BR: NOVATEC, 2008.

FRANCISCO JOSé CANDEIAS FIGUEIREDO, P. L. d. G. Acesso remoto em firewalls e topologia para gateways VPN. **Comissão Especial em Segurança da Informação e de Sistemas Computacionais**, [S.l.], 2001.

JAMES F. KUROSE, K. W. R. **Computer Network A Top-Dow Approach**. 6th.ed. USA: NOVATEC Co., Inc., 2016.

OLIVEIRA, R. R. Criptografia simétrica e assimétrica: os principais algoritmos de cifragem. **7a Conferencia sobre Redes de Computadores**, [S.l.], 2012.

STEPHEN NORTHCUTT LENNY ZELTSER, S. W. K. K. R. W. R. **Inside Network Perimeter Securiy**. 2th.ed. USA: NOVATEC Co., Inc., 2005.