MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

CAP EDSON BARBOSA DE SOUZA

DETECÇÃO DE ATAQUES LR DDOS *SLOWLORIS* USANDO ENTROPIA DE SHANNON

Rio de Janeiro
2018

INSTITUTO MILITAR DE ENGENHARIA

CAP EDSON BARBOSA DE SOUZA

DETECÇÃO DE ATAQUES LR DDOS *SLOWLORIS* USANDO ENTROPIA DE SHANNON

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador: Prof. Anderson Fernandes Pereira dos Santos - D.Sc.

Rio de Janeiro
2018

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 - Praia Vermelha
Rio de Janeiro - RJ CEP 22290-270

004 de Souza, Edson Barbosa
S729d DETECÇÃO DE ATAQUES LR DDOS *slowloris* usando entropia de Shannon / Edson Barbosa de Souza, orientado por Anderson Fernandes Pereira dos Santos - Rio de Janeiro: Instituto Militar de Engenharia, 2018.

80p.: il.

Dissertação (mestrado) - Instituto Militar de Engenharia, Rio de Janeiro, 2018.

1. Curso de Sistemas e Computação - teses e dissertações. 1. DDoS. 2. Ataques de Negação. I. dos Santos, Anderson Fernandes Pereira . II. Título. III. Instituto Militar de Engenharia.

INSTITUTO MILITAR DE ENGENHARIA

CAP EDSON BARBOSA DE SOUZA

DETECÇÃO DE ATAQUES LR DDOS *SLOWLORIS* USANDO ENTROPIA DE SHANNON

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador: Prof. Anderson Fernandes Pereira dos Santos - D.Sc.

Aprovada em 02 de Fevereiro de 2018 pela seguinte Banca Examinadora:

Prof. Anderson Fernandes Pereira dos Santos - D.Sc. do IME - Presidente

---

Prof^{a}. Raquel Coelho Gomes Pinto - D.Sc. do IME

Prof. Claudio Miceli de Farias - D.Sc. da UFRJ

Rio de Janeiro
2018

Ao Instituto Militar de Engenharia, alicerce da minha formação e aperfeiçoamento.

AGRADECIMENTOS

Ofereço as minhas respeitosas reverências aos pés de lótus de meu mestre espiritual iniciador Sri Sripad Bhaktivedanta Siddhanti Maharaj, aos pés de lótus de Sri Srimad Bhakti Vijñana Bharati Gosvami Maharaja, dos mestres espirituais predecessores e aos pés de todos os vaisnavas, como Swami Bhaktivedanta Mahavir e Maha Visnu Das. Ofereço as minhas respeitosas reverências ao Senhor Krsna Caitanya e ao Senhor Nityananda, e também a Advaita Acarya, Gadadhara, Srivasa e aos demais associados. Ofereço minhas respeitosas reverências a Srimati Radharani e Sri Krsna.

Agradeço ao meu orientador TC Anderson por sua dedicação e paciência em ler diversas vezes o trabalho e sugerir modificações.

Ao Instituto Militar de Engenharia que propiciou a minha formação intelectual.

A minha família, especialmente a minha esposa Valeria que sempre me apoiou.

“Ciência sem religião é materialismo, religião sem ciência é fanatismo . ”

A.C BHAKTIVEDANTA SWAMI PRABHUPADA

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABREVIATURAS

ACK	-	Número de Reconhecimento
C & C	-	Comando e Controle
CAIDA	-	*Center for Applied Internet Data Analysis*
CAPTCHA	-	*Completely Automated Public Turing test to tell Computers and Humans Apart*
CIAC	-	*Computer Incident Advisory Capability*
CPU	-	Unidade Central de Processamento
CR	-	*Carriage Return*
CTF	-	*Capture The Flag*
DARPA	-	*Defense Advanced Research Projects Agency*
DNS	-	*Domain Name System*
DoS	-	*Denial of Service*
DDoS	-	*Distributed Denial of Service*
EUA	-	Estados Unidos da América
GB	-	*Gigabyte*
HD	-	Disco Rígido
HOIC	-	*High Orbit Ion Cannon*
HTTP	-	*Hypertext Transfer Protocol*
HTTPS	-	*Hypertext Transfer Protocol Secure*
IME	-	Instituto Militar de Engenharia
ICMP	-	*Internet Control Message Protocol*
IDS	-	*Intrusion Detection System*
I/O	-	Entrada e Saída
IOT	-	*Internet of Things*
IP	-	*Internet Protocol*
IRC	-	*Internet Relay Chat*
ISN	-	Número de Sequência Inicial
ISS	-	*Internet Information Services*
ITU	-	*International Telecommunication Union*
JSON	-	*JavaScript Object Notation*
KDD	-	*Knowledge Discovery and Data Mining*
LAN	-	*Local Area Network*

LF - *Line Feed*
LOIC - *Low Orbit Ion Cannon*
LR - *Low Rate*
MB - *Megabyte*
MSS - *Maximum Segment Size*
MTU - *Maximum Transfer Unit*
NFV - *Network Functions Virtualization*
NTP - *Network Time Protocol*
OSI - *Open System Interconnection*
PCAP - *Packet Capture*
QOS - *Quality of Service*
RAM - *Random Access Memory*
RFC - *Request for Comments*
RoQ - *Reduction of Quality*
RTO - *Retransmission Timeout*
RUDY - *R-U-Dead-Yet*
SE - Seção de Ensino
SMB - *Server Message Block*
SSL - *Secure Socket Layer*
TB - *Terabyte*
TCB - *Transmission Control Block*
TFN - *Tribe Flood Network*
THC - *The Hackers Choice*
TLS - *Transport Layer Security*
TCP - *Transmission Control Protocol*
TTL - *Time To Live*
UDP - *User Datagram Protocol*
XML - *eXtensible Markup Language*

RESUMO

Uma das principais dificuldades no estudo de ataques de negação de serviço é a ausência de *datasets* completos e publicamente acessíveis. Os disponíveis, em geral, não contém os ataques atuais, como os do tipo *slow* HTTP. Existem apenas poucos *datasets* com ataques *LR DDoS* disponíveis publicamente. Ataques *LR DDoS* (*Low Rate Distributed Denial of Service*) exploram protocolos como HTTP a fim de tornar um servidor indisponível aos usuários legítimos, porém com um volume de tráfego em geral menor que o empregado em ataques *flooding*. Exemplos de ataques LR DDoS são o *slowloris* e o *sockstress*. Ataques do tipo *LR DDoS* geram tráfego muito similar ao legítimo, tornando a sua detecção um desafio.
Ao longo do texto, são descritos os experimentos que foram realizados para gerar o *dataset* de ataques distribuídos de negação de serviço (DDoS) *slowloris*, *sockstress* e comparadas algumas taxonomias de ataques *DDoS*.

A entropia, que é a medida da incerteza relacionada com uma variável aleatória, mede a informação média associada às observações da variável e é um conceito da Teoria da Informação muito utilizado para a detecção de ataques *DDoS*. Dentre as diversas métricas para o cálculo de entropia, destacam-se a de *Hartley*, a de *Shannon*, a de *Renyi* e a de *Kullback–Leibler* ou distância de *Kullback–Leibler*. As mais efetivas para a detecção de tráfego anormal são a entropia de *Shannon* e a de *Kullback–Leibler*. O objetivo deste trabalho é propor um método para a detecção de ataques *LR DDoS slowloris* a um servidor *web* utilizando o cálculo da entropia de *Shannon* (entropia de IP origem ou entropia de IP destino) e mais duas métricas criadas nesta pesquisa: medidas de taxa de envio de pacotes vazios por segundo (PVS) e taxa de recebimento dos caracteres delimitadores de fim de cabeçalho do protocolo HTTP, FCS (Fim de Cabeçalho por Segundo).

ABSTRACT

One of the main difficulties in the study of denial of service attacks is the absence of publicly accessible complete datasets. Those available, in general, do not contains the current attacks, like slow HTTP. There are only few datasets publicly available with DDoS LR attacks. Low Rate Denial of Service attacks exploit protocols such as HTTP in order to make a server unavailable to legitimate users, but with a volume of traffic generally lower than that employed in flooding attacks. Examples of LR DDoS attacks are slowloris and sockstress.
LR DDoS attacks generate a lot of traffic similar to the legitimate one, making its detection a challenge.
There are only few datasets publicly available with DDoS LR attacks. The experiments performed to generate the distributed denial of service attack dataset with slowloris, sockstress are described and some taxonomies of DDoS attacks are compared.

Entropy, which is the measure of uncertainty related to a random variable, measures the average information associated with the observations of the variable and it is a concept of the Information Theory that is much used for the detection of DDoS attacks. There are several metrics for calculating entropy, we highlight the Hartley entropy, the Shannon entropy, the Renyi entropy, and the Kullback-Leibler entropy or Kullback-Leibler distance. The most effective for detecting abnormal traffic are the Shannon entropy and the Kullback-Leibler entropy. The goal of this thesis is to propose a method for the detection of LR DDoS slowloris attacks to a web server using the Shannon entropy calculation (source IP entropy or destination IP entropy) and two more metrics created in this research: rate measures sending packets with empty payload (PVS) and receive rate of the HTTP protocol header end delimiter characters (FCS).

1 INTRODUÇÃO

Os países cada vez mais dependem de Tecnologia da Informação para prover os mais diversos serviços. No Brasil, a Segurança Nacional inclui a redução de vulnerabilidades dos sistemas relacionados à Defesa Nacional contra ataques cibernéticos (JUNGMANN, 2012). Segundo (MANDARINO JUNIOR; CANONGIA, 2010), "a Segurança Cibernética, desafio do século XXI, vem se destacando como função estratégica de Estado, e essencial à manutenção das infraestruturas críticas de um país, tais como Energia, Defesa, Transporte, Telecomunicações, Finanças, da própria Informação, dentre outras".

Dittrich et al. (2004) afirmam que o propósito do ataque *DoS* (*Denial of Service*: Negação de Serviço) é evitar que máquinas da rede sirvam os usuários legítimos: nenhum dado é roubado. A vítima não mais consegue servir seus clientes por estar sendo mantida ocupada pelo ataque (DITTRICH et al., 2004). Há diversas motivações para ataques *DoS*, segundo Dittrich et al. (2004), Mirkovic e Reiher (2004):

- políticas e diferenças ideológicas;
- crimes como extorsões e chantagens: ameaça-se lançar o ataque caso não receba certa quantia em dinheiro;
- ganhos financeiros e espionagem: prejudicar competidores no mercado;
- vingança;
- busca por prestígio na comunidade *hacker*;e
- guerra: destruir infraestrutura civil e militar.

Um ataque *DDoS* (*Distributed Denial of Service* : Ataque Distribuído de Negação de Serviço) frequentemente se baseia no uso de agentes de *software* denominados *bots* que são instalados de forma ilegítima em um grande número de máquinas infectadas por um *malware* (RAI; CHALLA, 2016), (BOTEANU; FERNANDEZ, 2013), (SELVARAJ et al., 2016). Os elementos que participam do ataque são chamados de zumbis e atuam sob as ordens de um nó mestre denominado *botmaster*. (SILVA et al., 2013), (ZLOMISLIĆ et al., 2017) definem *botnet* como uma rede constituída de máquinas infectadas por um *malware* e que são preparadas para a execução de atividades ilegais em escala mundial.

Os *bots* ou zumbis recebem as ordens do mestre chamado *botmaster* ou *herder* através de uma outra camada denominada Comando e Controle (C & C). A figura 1.1 exibe a configuração típica de uma *botnet* (SILVA et al., 2013). As setas simbolizam as trocas de mensagens entre os *bots* e o Comando e Controle, e entre o *botmaster* e o Comando e Controle.

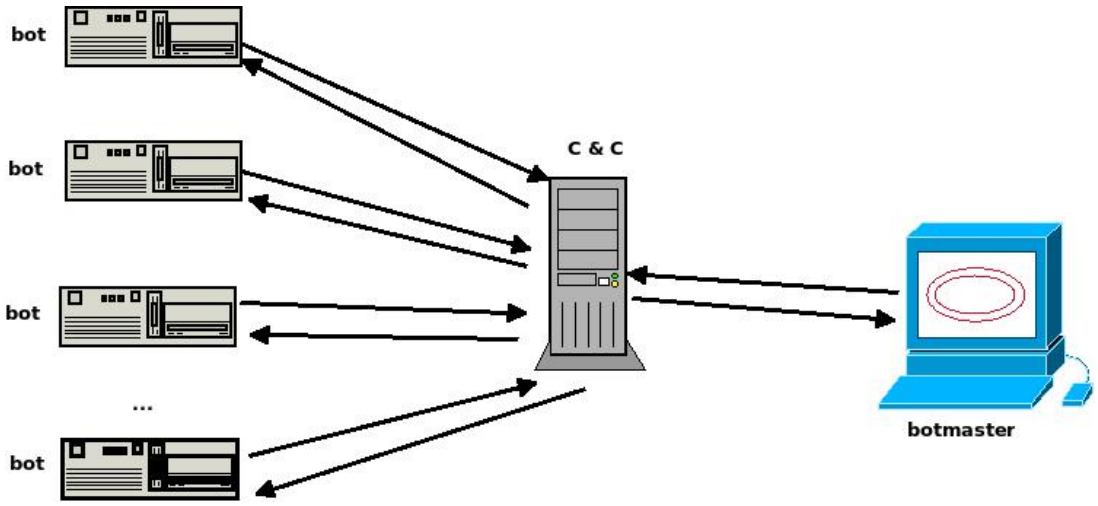

FIG. 1.1: Estrutura de uma *botnet* típica: baseado em Silva et al. (2013)

Um ataque *DDoS* é uma especialização do ataque *DoS* e visa indisponibilizar algum recurso computacional a usuários legítimos, normalmente através da inundação da rede (*flooding*), impedindo qualquer serviço, ou interrompendo algum serviço específico, o que além de prejuízos financeiros, pode macular a imagem de uma instituição. Os ataques distribuídos de negação de serviço são costumeiramente executados por uma grande rede de dispositivos remotamente controlados que se encontram distribuídos geograficamente (ZARGAR et al., 2013).

Em 1999, *Computer Incident Advisory Capability* (CIAC) dos Estados Unidos reportou o primeiro ataque *DDoS* (CRISCUOLO, 2000). Em 2016, o ataque *DDoS* mais longo demorou 197 horas (ou 8,2 dias) (KASPERSKY, 2016). No final de Outubro de 2016, um massivo ataque *DDoS* aos servidores de *DNS* nos EUA, afetou serviços como *Twitter*, *Spotify*, *Reddit*, *Airbnb* e *Vox* (GLOBO, 2016). Segundo o Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br), o número de ataques *DDoS* no país cresceu 138% em 2016.

Enquanto que para usuários comuns um ataque *DDoS* pode causar alguns inconvenientes ou até a redução de ganhos pecuniários, um ataque massivo a um sistema de *smart grid* pode levar a um *blackout* (BROOKS et al., 2016). Segundo Santos et al. (2010), "*estes ataques são baseados, principalmente, no consumo de recursos disponíveis*

dos servidores alvos, como o canal de comunicação e memória."

Dentre os diversos tipos de ataques cibernéticos que ocorreram em 2016, um dos que mais prejudicaram os negócios foram os ataques de negação de serviço (RADWARE, 2016a).

Na primeira semana de 2017, a maior empresa britânica de hospedagem de páginas eletrônicas, a 123-Reg, sofreu um massivo ataque *DDoS* (SOCPEDIA, 2017). Entre 11 e 13 de Janeiro de 2017, o banco *Lloyds* no Reino Unido sofreu ataques *DDoS* que resultaram prejuízo de 2,5 milhões de libras (NARINDER, 2017).

1.1 OS ATAQUES *LR DDOS*

Uma variante importante dos ataques *DDoS* são os do tipo *LR* (*Low Rate*: Baixas Taxas), em que o alvo é a camada de aplicação, visando protocolos como o HTTP. Mecanismos de defesa como *IPS* e *firewalls* não são muito efetivos, pois operam normalmente na camada de rede. Um exemplo de ataque *LR DDoS* é o *slowloris*, que funciona pelo envio de requisições HTTP com cabeçalho incompleto nas quais o servidor *web* ficará esperando o envio do restante.

O ataque *slowloris* foi escolhido para objeto da dissertação em função de ainda ser funcional (não é específico de um tipo de servidor como o Apache, mas da forma como o protocolo HTTP foi especificado em sua RFC) e, em função disso, surgem inúmeras variações do mesmo. Em (GRAHAM, 2017), os pesquisadores Sean Dillon e Zach Harding anunciaram o *SMBloris* um ataque similiar ao *slowloris* mas visando o protocolo SMB (*Server Message Block*) e, possivelmente, o *software* Samba, permitindo, por exemplo, consumir todos os recursos de uma placa *Raspberry Pi* mesmo em um grande servidor corporativo .

1.2 ATAQUES *DDOS* E A INTERNET DAS COISAS

Com a quantidade cada vez maior de diversos dispositivos com conexão à internet, tais como, máquinas de lavar, geladeiras, *smartphones*, televisores, a preocupação com a segurança deixou de ser privilégio apenas de computadores. Há um grande número de equipamentos (*IoT*: Internet das coisas) inseguros e com alto poder computacional que têm potencial de tornar-se parte de uma grande *botnet* (BERTINO; ISLAM, 2017).

O ITU (*International Telecommunication Union*) formulou a seguinte definição de *IoT* (ASWATHY; MALARVIZHI, 2018):

"*Uma infraestrutura global para a sociedade da informação, permitindo serviços avançados por meio da interligação das coisas (físicas e virtuais) baseadas na interoperabilidade das tecnologias de informação e de comunicação existentes e em evolução.*"Como exemplos de aplicações de *IoT* pode-se citar o monitoramento no transporte de cargas perigosas e químicas, a distribuição de energia (*smart grid*), os sistemas de informação de tráfego e controles de estoque.

Os ataques *slow DDoS*, como o *slowloris*, exploram o funcionamento normal do protocolo HTTP e, assim, os dispositivos *IOT* podem ser vulneráveis. Em 2017, houve um aumento de 91% de ataques *DDoS* em função de dispositivos *IoT* (RAYOME, 2017).

1.3 MOTIVAÇÃO

Os ataques *LR DoS* (*Low Rate Denial of Service*) exploram protocolos como HTTP e HTTPS para enviar o tráfego a vítima, causando uma sobrecarga de CPU, memória e outros recursos, tornando o servidor indisponível. O volume de tráfego necessário a tal negação de serviço é, em geral, menor que o empregado em ataques que utilizam inundação (BHATTACHARYYA; KALITA, 2016).

Um atacante é capaz de simular tráfego de rede legítimo a fim de confundir os algoritmos de detecção, como, por exemplo, falsificando o IP de origem de um pacote (IP *Spoofing*) (YU, 2014). Os profissionais de segurança da informação estão sempre tentando implantar medidas que se antecipem aos atos dos agentes mal intencionados porém, contramedidas a usuários maliciosos podem impactar negativamente na qualidade de serviço (*QoS: quality of service*) aos usuários legítimos (BOTEANU; FERNANDEZ, 2013).

(FEINSTEIN et al., 2003) fez um prognóstico da dificuldade que os administradores de redes teriam em detectar ataques *DDoS*, distinguindo-os de tráfegos legítimos, em função da evolução das ferramentas de ataque que permitissem inundações de rede com taxas de pacotes baixas.

Há diversas abordagens para a detecção de ataques *DDoS*, como a utilização de *Hurst Parameter* e lógica *fuzzy* (PHARANDE et al., 2015), *Decision Tree machine learning* (VUONG et al., 2015), modelos estatísticos de *Bernoulli* e *Multinomial* (RMAYTI et al., 2014), visão computacional (o tráfego de rede é tratado como imagem) (TAN et al., 2015), algoritmos genéticos e redes neurais (BARATI et al., 2014), entropia e uso de CPU ou I/O para identificar o ataque em um *datacenter* na nuvem (CAO et al., 2015); (DAVID; THOMAS, 2015), classificador *REPTree (Reduced Error Pruning Tree)* (KATKAR; BHATIA, 2013), comparação do número de pacotes com um *threshold* (KSHIRSAGAR et al., 2016).

(ÖZÇELIK; BROOKS, 2015),(DAVID; THOMAS, 2015),(HAMDI; BOUDRIGA, 2007), (GAVRILIS; DERMATAS, 2005),(FEINSTEIN et al., 2003) indicam que os métodos estatísticos são uma abordagem efetiva no problema de detecção de ataques *DDoS*. Métodos de detecção de ataques *DDoS* baseados em entropia são relevantes e, em uma rede em funcionamento normal, seus valores são suaves (DAVID; THOMAS, 2015).

1.4 CARACTERIZAÇÃO DO PROBLEMA

A detecção de um ataque *LR DDoS* como o *slowloris* ou o *sockstress* é um desafio (LUO et al., 2014), (KHARAT; NAIK, 2013), (KUZMANOVIC; KNIGHTLY, 2006) , (KUZMANOVIC; KNIGHTLY, 2003).

O ataque *sockstress* é um ataque *DDoS* que explora uma vulnerabilidade no protocolo TCP com relação ao tamanho da janela. O cliente malicioso estabelecerá a conexão TCP com tamanho de janela igual ou próximo a zero e, o servidor alvo continuamente enviará pacotes de descoberta do tamanho de janela a fim de aumentar seu valor e iniciar efetivamente a comunicação, mantendo a conexão aberta por tempo indeterminado.

No ataque *slowloris*, o atacante envia os cabeçalhos HTTP sem os caracteres indicativos de seu final (`\r\n\r\n`), repetidamente de modo a manter a conexão ativa. O servidor mantém a conexão aberta aguardando o restante do cabeçalho, que o atacante nunca enviará por completo. Sem precisar de muita largura de banda, um atacante pode abrir muitas conexões e sobrecarregar o servidor *web*.

1.5 OBJETIVO

(SHARMA et al., 2015) analisou como os valores de entropia de diferentes atributos (IP de origem, IP de destino, *flags*, protocolo, porta de origem e porta de destino) evoluem quando ocorrem ataques na camada de rede por inundação (TCP SYN *flood*, *Smurf* e UDP *flood*). O objetivo do presente trabalho é detectar o ataque *slowloris* a um servidor *web* utilizando os valores de entropia de *Shannon* (entropia de IP origem ou entropia de IP destino) e medidas de taxa de envio de pacotes com *payload* vazio (PVS) e taxa de recebimento dos caracteres delimitadores de fim de cabeçalho (FCS) do protocolo HTTP, comparando-as no tráfego de fundo e no tráfego de ataque.

1.6 CONTRIBUIÇÕES

As principais contribuições esperadas nesta dissertação são:

a) *dataset* contendo os ataques LR DDoS *slowloris* e *sockstress* e a análise crítica do experimento realizado para a sua obtenção;

b) um método para a detecção do ataque *slowloris* utilizando a entropia de *Shannon*.

É apresentado o novo *dataset* de ataques LR *DDoS* desenvolvido, que vem preencher uma lacuna na área, já que outros *datasets* não apresentam os ataques *slowloris* e *sockstress*.

1.7 ORGANIZAÇÃO DO TRABALHO

A presente dissertação está organizada como segue: após a introdução, são apresentados no Capítulo 2 o funcionamento dos protocolos TCP e HTTP, as suas vulnerabilidades e ataques e no Capítulo 3, os principais trabalhos relacionados. O Capítulo 4 inicia com uma abordagem de aspectos da Teoria da Informação e da Entropia. No Capítulo 5 é apresentada a proposta de um algoritmo para a detecção de ataques *LR DDoS slowloris* e no Capítulo 6 são analisados os resultados obtidos. O Capítulo 7 apresenta as conclusões e os trabalhos futuros. Nos Apêndices, são apresentados os principais *datasets* para pesquisas de ataques *DDoS*, sua comparação e é descrito o experimento executado para atingir os objetivos desta dissertação.

2 OS PROTOCOLOS TCP E HTTP: FUNCIONAMENTO, VULNERABILIDADES E ATAQUES

"Segurança da informação é a proteção da informação contra vários tipos de ameaças para garantir a continuidade do negócio, minimizar o risco ao negócio, maximizar o retorno sobre os investimentos e as oportunidades de negócio"(ABNT, 2013). É uma área que tem por finalidade proteger as informações de valor para um indivíduo ou organização, assegurando, assim, as seguintes características (STALLING, 2016):

a) confidencialidade: a informação somente será acessada por pessoas autorizadas;

b) integridade: impedir a modificação ou destruição de informações de forma não-autorizada;

c) disponibilidade: sempre que solicitadas as informações estão acessíveis.

Uma ameaça à Segurança da Informação é um elemento capaz de restringir quaisquer dos três princípios mencionados anteriormente e, assim, de forma abrangente, pode-se afirmar que tem-se uma negação de serviço sempre que algum tipo de recurso deixa de ser acessível por qualquer razão.

Quando um ataque é proveniente de muitos dispositivos diferentes controlados por um agente e cujo objetivo é impedir o acesso de usuários legítimos, tem-se um ataque *DDoS*(*Distributed Denial of Service*: Ataque Distribuído de Negação de Serviço).

(MIRKOVIC; REIHER, 2004) define ataque distribuído de negação de serviço (*DDoS*) como aquele em que múltiplos equipamentos são utilizados em um ataque *DoS*. Muitos ataques *DoS* exploram falhas em *softwares* fazendo-os entrar em *loop*. Inúmeros ataques são assimétricos: um atacante com pequenos recursos (número reduzido de máquinas), enviando pacotes mal-formatados podem causar grandes estragos.

Algumas características comuns a ambientes computacionais sob um ataque de negação de serviço são (RAI; CHALLA, 2016):

- lentidão na rede;
- *websites* indisponíveis e
- desconexões de redes.

Um desafio na detecção de ataques é que, com frequência, os atacantes alteram as suas técnicas afim de ofuscar as equipes de segurança da informação (HOQUE et al., 2016) (MIAO et al., 2015).

2.1 O PROTOCOLO TCP E O ATAQUE *SOCKSTRESS*

O TCP (*Transmission Control Protocol*) é um exemplo de um protocolo vulnerável: é implicitamente assumido que os usuários, em geral, não são maliciosos (KUZMANOVIC; KNIGHTLY, 2003) e foi projetado especificamente para oferecer um fluxo de *bytes* fim a fim confiável em uma inter-rede não confiável (TANENBAUM, 2011).

O *sockstress* é um ataque *LR DDoS* que surgiu em 2008 e explora uma vulnerabilidade na fase de negociação do tamanho da janela do protocolo TCP (HAVOC, 2012), (CENTRE, 2009). O atacante completa o TCP *handshake* de três vias com o valor de janela igual ou próximo a zero e o servidor mantendo a conexão aberta, tenta negociar um valor de janela maior para a transferência de dados. Como o cliente é malicioso, abre, intensionalmente, inúmeras conexões semelhantes de forma a esgotar os recursos (CPU, memória, etc) do servidor alvo.

No processo de estabelecimento de conexão por meio do *handshake* de três vias, enquanto o cliente não enviar o segmento *ACK*, tem-se uma conexão *half-open* (semi-aberta) em que os recursos necessários estarão disponíveis até o *timeout*.

Um atacante estabelece uma conexão TCP com servidor alvo remetendo o tamanho da janela com valor nulo no último ACK, forçando, assim, ao servidor definir o tamanho da janela TCP como 0 *bytes* (RADWARE, 2016a). O valor do tamanho de janela nulo significa que não há mais espaço no *buffer* e que o emissor deve parar de enviar dados até que se avise o contrário. Assim sendo, o servidor tentará negociar continuamente um tamanho de janela maior, enviando pacotes de descoberta (*window probe*) ao cliente e, como este, não é bem intencionado , manterá a variável tamanho da janela TCP inalterada, forçando o servidor a manter a conexão aberta por tempo indeterminado. Por abrir muitas conexões TCP *half-open*, o atacante é capaz de exaurir todo o espaço disponível nas tabelas TCB (*Transmission Control Block*), impedindo que os clientes legítimos sejam atendidos, ocasionando a negação de serviço.

2.2 O PROTOCOLO HTTP E O ATAQUE *SLOWLORIS*

O ataque *slowloris* surgiu em 2009 (CROSS, 2011). O grupo *Anonymous* utilizou uma ferramenta homônima a fim de lançar o ataque em servidores Apache (THOMSON; CON-

RAD, 2011).

O protocolo HTTP (*Hypertext Transfer Protocol*), cuja versão mais recente é a 2.0, está definido na RFC 7540 e é compatível com a versão anterior 1.1, sendo decidido durante a fase de negociação qual das duas versões será utilizada (DIMITROVA; MILEVA, 2017). Uma das diferenças entre as duas versões é que no HTTP 1.1 os cabeçalhos são enviados em modo texto e no HTTP 2.0 em modo binário, enviando os *headers* comprimidos com o algoritmo HPACK por padrão, diminuindo o volume de dados trafegados (DIMITROVA; MILEVA, 2017). Na documentação do protocolo HTTP (BELSHE et al., 2015) é especificado que o servidor *web* aguarde a recepção completa da requisição HTTP, que é caracterizado por (`\r\n\r\n`) antes de efetuar o processamento, o que permite alguns tipos de ataques específicos, como os *Slow* HTTP (*slowloris*, *slow body* e *slow read*). De acordo com a RFC 2616, o fim de linha no cabeçalho do protocolo HTTP é marcado pelos caracteres "CR LF ou `\r\n"` e o fim do cabeçalho (início da mensagem) por " `\r\n\r\n`". Em uma requisição HTTP normal, o cliente envia HTTP GET ao servidor e este responde com o código 200, que significa sucesso na conexão, enviando em seguida os dados solicitados.

Os ataques do tipo *Slow* HTTP têm como alvo a camada de aplicação. Quando uma requisição HTTP é incompleta ou a taxa de transferência é muito baixa, o servidor *web* mantém a conexão aberta aguardando o restante dos dados. Há três tipos de técnicas utilizadas pelos ataques *Slow* HTTP (PARK et al., 2015):

- Enviar o cabeçalho da requisição lentamente;
- Ler a resposta lentamente;e
- Enviar o corpo da resposta da requisição lentamente.

O primeiro tipo é denominado *slowloris* (ou *Slow* HTTP *Headers* ou *Slow* HTTP GET); o segundo é o *Slow Read* e o terceiro *Slow body* ou RUDY.

No ataque *slowloris*, o atacante envia os cabeçalhos HTTP sem os caracteres indicativos de seu final (`\r\n\r\n`), repetidamente de modo a manter a conexão ativa. O servidor mantém a conexão aberta aguardando o restante do cabeçalho, que o atacante nunca enviará por completo. Sem precisar de muita largura de banda, um atacante pode abrir muitas conexões e sobrecarregar o servidor *web*.

Na figura 2.1 é exibido o cabeçalho HTTP em uma conexão normal, nota-se o "`\r\n\r\n`" indicando o fim do header e início dos dados requisitados. A figura 2.2 mostra

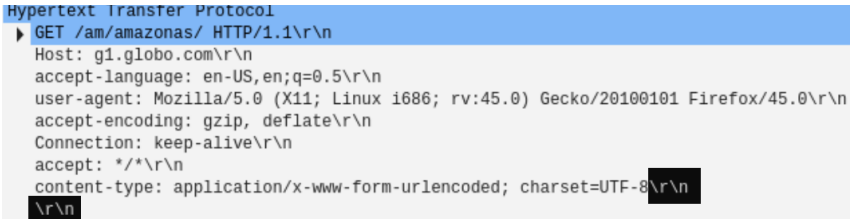

FIG. 2.1: cabeçalho em uma conexão HTTP normal

o cabeçalho HTTP durante um ataque *slowloris*. Observa-se apenas um "`\r\n`" de forma a manter a conexão ativa.

FIG. 2.2: cabeçalho em um ataque *slowloris*

No ataque *Slow Read*, o *handshake* de três vias é completado de forma legítima mas o atacante define um valor de tamanho de janela muito pequeno, como por exemplo 700 *bytes*, ao enviar o segmento ACK (TAYAMA; TANAKA, 2017). Desta forma, a resposta é recebida do servidor de forma lenta. Quando o tamanho da janela está próximo a zero, o servidor para de enviar dados e mantém a conexão ativa. O servidor é indisponibilizado ao serem efetuadas um números de conexões concorrentes suficientes para esgotar seus recursos. Até o presente momento, não foram encontradas contramedidas eficientes contra o ataque *slow read* (TAYAMA; TANAKA, 2017).

O ataque *Slow body* ou RUDY é caracterizado pelo envio do HTTP *Post header* completo mas com o campo *content-length* (tamanho da página em *bytes*) maior do que o seu tamanho real e enviado lentamente (por exemplo, 1 *byte*/min) de forma a manter o servidor ocupado (JAZI et al., 2017).

2.3 ATAQUES *DDOS*

A detecção de ataques *DDoS* com base em análises estatísticas do tráfego de rede comumente calcula a entropia apenas do endereço IP de origem (BROOKS et al., 2016), (FEINSTEIN et al., 2003), (MA; CHEN, 2014), (CARL et al., 2006), (ÖZÇELIK; BROOKS, 2015).

Afim de lançar um ataque *DDoS* pode-se utilizar agentes de *software*, denominados *bots*, que são instalados de forma ilegítima em um grande número de equipamentos (RAI; CHALLA, 2016), (BOTEANU; FERNANDEZ, 2013), (SELVARAJ et al., 2016). Em

(SILVA et al., 2013), é definida como *botnet* uma rede formada por máquinas infectadas por um *malware* e que são preparadas para a execução de atividades ilegais em escala mundial. Os elementos participantes do ataque são chamados de zumbis e atuam sob as ordens de um nó mestre denominado *botmaster*.

Um exemplo de taxonomia de ataques *DDoS* é a seguinte (RAI; CHALLA, 2016):

(a) Ataques baseados em volume (*Volume based*): os recursos computacionais da vítima (CPU, *buffers*, memória, rede, etc) são exauridos por um grande volume de tráfego malicioso, impedindo, assim, o atendimento de usuários legítimos, caracterizando a negação de serviço. Uma *botnet* normalmente é utilizada para a geração do tráfego malicioso. O ataque ocorre na camada de rede, transporte ou aplicação. Exemplos: *SYN flood*, UDP *flood*, *smurf*.

(b) Ataques *DDoS* na aplicação (*app based*): diferentes aplicações são alvo do atacante para consumir os recursos da vítima. Exemplo de ataque é o HTTP *flood* (HTTP GET ou HTTP POST).

(c) Ataques *Slow Rate* e *Low Rate* (*SL/LR*): são os tipos mais difíceis de detectar, pois operam abaixo dos patamares de detecção e apresentam características similares ao tráfego legítimo (KUZMANOVIC; KNIGHTLY, 2006). Exemplos: *slowloris*, *sockstress*, RUDY.

3 O ESTADO DA ARTE

Na literatura, encontram-se diversas abordagens para a detecção de ataques *DDoS*, como a utilização de *Hurst Parameter* e lógica *fuzzy* (PHARANDE et al., 2015), *Decision Tree machine learning* (VUONG et al., 2015), modelos estatísticos de *Bernoulli* e *Multinomial* (RMAYTI et al., 2014), visão computacional (o tráfego de rede é tratado como imagem) (TAN et al., 2015), algoritmos genéticos e redes neurais (BARATI et al., 2014), entropia e uso de CPU ou I/O para identificar o ataque em um *datacenter* na nuvem (CAO et al., 2015); (DAVID; THOMAS, 2015), classificador *REPTree (Reduced Error Pruning Tree)* (KATKAR; BHATIA, 2013), comparação do número de pacotes com um *threshold* (KSHIRSAGAR et al., 2016). A tabela-resumo 3.1 ilustra os trabalhos.

TAB. 3.1: Detecção de ataques *DDoS* na literatura

Referência	Técnica	Observações
(SELVARAJ et al., 2016)	*roaming virtual honeypots*	Inspirada em colônias de formigas.
(PHARANDE et al., 2015)	*Hurst Parameter* e lógica *fuzzy*	Uso do *dataset* DARPA 1998.
(VUONG et al., 2015)	*Decision Tree machine learning*	Abordagem para veículos não-tripulados.
(RMAYTI et al., 2014)	Modelos de classificação *Bayesian*	*Bernoulli* e *Multinomial* com o simulador NS2.
(TAN et al., 2015)	visão computacional	Trata o tráfego como imagem e valida com o *dataset* KDD Cup 99.
(BARATI et al., 2014)	algoritmos genéticos (GA) e redes neurais (ANN)	GA para seleção e ANN para detecção.
(KSHIRSAGAR et al., 2016)	n^o de pacotes $>$ *threshold*	testado com *TCP SYN Flood*.
(CAO et al., 2015)	entropia	entropia e uso de CPU e rede para identificar *DDoS*.
(DAVID; THOMAS, 2015)	entropia	utiliza o conceito de *Fast Entropy*.
(BROOKS et al., 2016)	entropia	entropia do IP de origem; validação com UDP *flood*.
(MA; CHEN, 2014)	entropia	entropia dos endereços IP de origem e destino. Utiliza *dataset* DARPA 1998. Não valida com *LR DDoS*.

(BROOKS et al., 2016) demonstrou uma técnica para detecção de ataques *DDoS* usando a entropia do IP de origem do pacote e os experimentos são executados com o ataque UDP *flood*. É sugerida a possibilidade de utilizar a entropia de outros campos do cabeçalho do pacote IP.

(MA; CHEN, 2014) calcula a entropia dos endereços IP de origem e destino e emprega o *dataset* do projeto DARPA 1998, que também não contém ataques *LR DDoS*. (JIAN-QI et al., 2013) propõe um método de entropia dinâmica para detecção de ataques *DDoS* que é validado com o *dataset* DARPA 1999. As técnicas estão resumidas na tabela 3.2.

TAB. 3.2: Detecção de *DDoS*

Referência	Técnica	Variável de Interesse	Observações
(CAO et al., 2015)	entropia	entropia do vm *data-center*	entropia e uso de CPU, rede ou I/O para identificar *DDoS* em *data center* na nuvem.
(DAVID; THOMAS, 2015)	entropia	*fast entropy* do fluxo	ataque detectado qdo a diferença entre a entropia do fluxo a cada instante e sua média no intervalo é maior que um *threshold* .
(BROOKS et al., 2016)	entropia	entropia do cabeçalho do pacote	validação com UDP *flood*.
(MA; CHEN, 2014)	entropia	entropia dos endereços IP de origem e destino	Utiliza *dataset* DARPA 1998. Não valida com *LR DDoS*.
(FEINSTEIN et al., 2003)	entropia dinâmica	entropia do IP origem/dest	validado com o *dataset* DARPA 1999.

Várias pesquisas foram realizadas nos últimos anos com o objetivo de estudar a detecção de ataques *LR DDoS*. Os tráfegos *slow* HTTP são muito similiares aos produzidos por usuários legítimos, tornando difícil a sua identificação e detecção.

Em (SHAFIEIAN et al., 2015) são utilizadas florestas aleatórias a fim de construir classificadores para a detecção de tráfego *slow read*, enquanto que em (SHARMA, 2014) e (MONGELLI et al., 2015) empregam-se análise espectral para a detecção de ataques *LR DDoS*.

Em (ADI et al., 2016) é provado que o protocolo HTTP/2 é mais vulnerável a ataques *low rate* que o HTTP versão 1.1 e a abordagem de (BECKETT et al., 2017) é monitorar as consultas HTTP e construir um perfil de cada acesso individual, de forma a detectar

os ataques *slow* HTTP.

Em (ALHARBI et al., 2017) é proposto um *framework* em dois estágios utilizando a virtualização de funções de rede, a NFV (*Network Functions Virtualization*) para a detecção de ataques *LR DDoS* e *flooding*.

(CUSACK et al., 2016) elaborou um método de detecção e identificação de ataques *slow DDoS* com distância baseada em métricas de similaridade (distância Euclidiana), enquanto (KATKAR et al., 2015) apresentou a arquitetura de um IDS (*Intrusion Detection System*) para a detecção de ataques *DDoS* a servidores *web* utilizando processamento distribuído e classificador Naive bayes.

(SAIED et al., 2016) demonstrou um algoritmo de detecção baseado em redes neurais artificiais para separar os padrões de ataque e tráfego legítimo e (DURAVKIN et al., 2014) descreveu um modelo para a detecção de *LR DDoS* baseado em cadeias de Markov, teoria de filas e funções geradoras.

(HOQUE et al., 2017) utilizou uma medida de correlação denominada NaHiD, que é similar a correlação de *Pearson*, *Spearman* e *Kendall* para a detecção de ataques *DDoS* em tempo real.

Em (TRIPATHI; HUBBALLI, 2017) é proposto um esquema de detecção em duas fases, sendo uma de treinamento e outra de testes. Na fase de testes, em que são comparados o tráfego gerado na fase de treinamento com o tráfego atual, é utilizado o teste estatístico do qui-quadrado.

Em (D'CRUZE et al., 2017) e (SHARMA et al., 2015) é indicada a possibilidade de utilizar entropia de Shannon, Hartley ou Renyi para a detecção de ataques *LR DDoS*.

(SHARMA et al., 2015) analisou como os valores de entropia de diferentes variáveis (IP de origem, IP de destino, *flags*, protocolo, porta de origem e porta de destino) variam quando ocorrem ataques *flooding* (TCP *SYN flood*, *Smurf* e UDP *flood*). Pode-se questionar se semelhante resultado aplica-se ao ataque *LR DDoS sockstress*. Mostra graficamente a variação de entropia para diferentes tipos de ataques *DDoS*, como o TCP/UDP SYN *flooding* e *smurf*, criticando a utilização de entropia de endereços IP em função de *spoofing*. Como no ataque *slowloris* uma conexão é estabelecida por meio do *handshaking* de três vias do TCP, o presente trabalho utilizará medidas de entropia de endereços IP (origem e destino).

4 A ENTROPIA E A TEORIA DA INFORMAÇÃO

A Teoria da Informação é um ramo da Matemática que estuda a quantificação da informação (COVER; THOMAS, 2012).

Segundo (EPSTEIN, 1986):

"Informação é uma redução de incerteza, oferecida quando se obtém resposta a uma pergunta".

Supondo-se que um evento A ocorra com probabilidade $P(A)$, a informação associada a observação de sua ocorrência é definida como (BANERJEE, 2016):

$$P(A) = \log_2 \left[\frac{1}{P(A)} \right] = -\log_2[P(A)]$$

Claude Shannon partiu da Entropia da Mecânica Estatística transformando-a em uma medida de quantidade de informação. Em 1948 com a publicação de seu artigo " *The Mathematical Theory of Communication*", foi introduzida a ideia de que os componentes de um sistema de comunicação (as fontes de informação e os canais de comunicação) são elementos probabilísticos. A descrição do processo de comunicação segundo Shannon inclui agentes (o emissor e o receptor), recursos (o canal de comunicação e a codificação) e métodos (a codificação das mensagens em símbolos).

Shannon utilizou-se da expressão da entropia de Ludwig Boltzmann (1896) da Termodinâmica, que afirma que a entropia (S) é proporcional ao número de microestados existentes (W) nos átomos que formam um gás ideal, sendo K a constante de Boltzmann, 1.346×10^{-23}:

$$S = -Klog_2(W) \tag{4.1}$$

A entropia definida por Boltzmann é uma estatística que relaciona-se a uma quantidade de matéria e que foi baseada na proposição de Maxwell chamada Lei da Distribuição de Velocidades. A teoria quântica afirma que os átomos e as moléculas não se encontram em um estado qualquer, mas somente em estados estáveis discretos e a transição de um estado para outro envolve absorção ou emissão de energia. A contagem dos estados quantizados é a medida da entropia do sistema.

A entropia é a medida da incerteza relacionada com uma variável aleatória (BHUYAN et al., 2015) e mede a informação média associada às observações da variável. A unidade

de entropia é a mesma da medida de informação; ao utilizar-se o logaritmo na base 2, a unidade de medida é em *bits*. Ela se relaciona ao grau de desorganização existente na fonte de informação. A entropia é máxima para o caso em que o espaço amostral é equiprovável e é nula quando a probabilidade associada a um valor é igual a um (determinístico).

Na Teoria da Informação, existem inúmeras métricas para o cálculo de entropia, como a de *Hartley*, a de *Shannon*, a de *Renyi* e a de *Kullback–Leibler* ou distância de *Kullback–Leibler*. A entropia de *Shannon* e a de *Kullback–Leibler* são as mais efetivas para a detecção de tráfego anormal (BHUYAN et al., 2015), (SILVA; SALLES, 2015).

A expressão para a entropia S segundo Shannon é (SHANNON; WEAVER, 1998):

$$S = -\sum_{i=1}^{n} W \log_2(W) \tag{4.2}$$

Onde n é o número de elementos ou símbolos existentes e W são os dados extraídos do *dataset* desenvolvido nesta pesquisa (vide Apêndice), a ocorrência dos endereços IP's em um determinado intervalo de tempo:

São separados os IPs de origem e destino em cada *time slice* de 1 minuto e calculadas as respectivas entropias, conforme o algoritmo que será apresentado no capítulo 5. Um exemplo numérico de cálculo da entropia é apresentado no Apêndice 5 da Dissertação.

Pode-se definir a entropia da fonte, S, como a informação média, obtida pela ponderação de todas as suas ocorrências. Shannon definiu a entropia de um alfabeto como o valor médio do logaritmo do inverso da probabilidade de ocorrência dos símbolos.

Shannon formulou o conceito de Capacidade de um Canal: num canal de transmissão qualquer, a velocidade nominal do canal equivale à entropia máxima que ele pode transmitir por unidade de tempo.

Observe que a Equação 4.2 é uma média ponderada dos logaritmos das probabilidades, na qual os pesos são as probabilidades dos valores da variável aleatória, o que sugere que S pode ser interpretada como o valor esperado da variável aleatória que assume o valor $-\log_2(W)$, com probabilidade W. A Equação 4.2 será utilizada para medir o grau de concentração de distribuições de probabilidade dos endereços IP, tanto de origem quanto de destino, calculadas em fluxos de tempo de um minuto.

4.1 PROPRIEDADES DA ENTROPIA

O valor máximo da entropia ocorre em eventos igualmente prováveis, isto é, quando nada se sabe sobre um conjunto de eventos, ou sobre qual mensagem foi gerada. Assumir a

distribuição uniforme fornece a maior quantidade de informação, o que corresponde ao nível mais elevado de incerteza. A entropia é invariante em relação à ordem dos eventos e não negativa, assim, o particionamento de um evento em vários outros não pode reduzir a entropia do sistema.

4.2 ENTROPIA CONJUNTA E ENTROPIA CONDICIONAL

Seja X uma variável aleatória com alfabeto χ e distribuição de probabilidade p(x) = Pr(X=x), x $\in \chi$ e Y uma variável aleatória com alfabeto γ e distribuição de probabilidade p(y) = Pr(Y=y), y $\in \gamma$.

A entropia conjunta de duas variáveis aleatórias X e Y, com distribuição de probabilidade conjunta $p(x, y)$ é:

$$S(X, Y) = -\sum_{x}\sum_{y} p(x, y) \log_2 \left[p(x, y)\right]$$

A entropia condicional $S(Y/X)$ é dada por:

$$S(Y/X) = -\sum_{x}\sum_{y} p(x, y) \log_2 \left[p(y/x)\right]$$

Pode-se escrever a entropia conjunta em função da entropia condiconal:

$$S(X, Y) = S(X) + S(Y/X) = S(Y) + S(X/Y)$$

4.2.1 MÉTRICAS PARA O CÁLCULO DA ENTROPIA

As principais métricas para o cálculo da entropia são a entropia de *Hartley*, a entropia de *Shannon*, a entropia de *Renyi* e a entropia de *Kullback–Leibler* ou distância de *Kullback–Leibler*.

A entropia de *Renyi* de ordem α é uma generalização da entropia de *Shannon*. Seja P uma distribuição de Probabilidade discreta, com P = p_1,p_2,p_3, ...,p_n, isto é, $\sum_{i=1}^{n} p_i = 1$, $p_i \geq 0$. A entropia de *Renyi* de ordem α é definida como:

$$S_\alpha(x) = \frac{1}{1-\alpha} \log_2(\sum_{i=1}^{n} p_i^\alpha) \tag{4.3}$$

Com $\alpha \geq 0$, $\alpha \neq 1$, $p_i \geq 0$. Se $p_1 = p_2 = p_3 = ... = p_n$, tem-se a entropia de *Hartley*, que é o valor máximo da entropia:

$$S_0(x) = \log_2 n \tag{4.4}$$

A entropia de *Renyi* tende a entropia de *Shannon* quando $\alpha \to 1$:

$$S_1(x) = -p_i \log_2(\sum_{i=1}^{n} p_i) \tag{4.5}$$

A distância de *Kullback–Leibler*, chamada também de entropia relativa ou divergência, é uma medida da distância entre duas distribuições de probabilidade: mede a ineficiência em assumir que a distribuição de probabilidade dos símbolos da fonte é q, quando a verdadeira distribuição é p. Quanto maior o valor da entropia relativa, maior será a distância entre a distribuição real e a distribuição que foi arbitrada. A expressão para a entropia de *Kullback–Leibler* é:

$$D(P \parallel Q) = \sum_{i=1}^{n} p_i \log_2 \frac{p_i}{q_i} \tag{4.6}$$

A entropia relativa é sempre não-negativa e não é uma distância no espaço euclidiano, uma vez que $D(P \parallel Q) \neq D(Q \parallel P)$.

Segundo (SILVA; SALLES, 2015), a entropia de Shannon permite verificar a dispersão de valores, enquanto a Divergência adequa-se a comparação entre duas distribuições consecutivas, de modo a identificar alterações bruscas.

Comparar os valores da entropia em um conjunto de pacotes durante um intervalo de tempo permite detectar variações em sua aleatoriedade (YU et al., 2011). Em uma situação em que não há ataques, a entropia de diversos campos do *header* de um pacote assume valores em um intervalo pequeno; quando acontece um ataque, há uma aumento dos seus valores. Da definição apresentada de entropia, espera-se que quanto mais organizado, menos randômico um tráfego de rede, maiores serão as chances de um ataque ter ocorrido e diferentemente de diversas técnicas estatísticas, não exige a marcarção de pacotes, sendo, assim, mais simples de aplicar em cenários reais. A entropia de *Shannon* permite avaliar o grau de concentração de determinadas distribuições de probabilidade dos endereços IP, tanto de origem quanto de destino, por exemplo, para fluxos de pacotes amostrados durante um intervalo de tempo.Quando ocorrem ataques distribuídos de negação de serviço, nota-se a presença de vários fluxos com valores diversificados para o IP de origem e um mesmo valor para o IP de destino, implicando em um distribuição dispersa para os endereços IP de origem e concentrada para os endereços IP de destino. Em função do baixo custo computacional envolvido no cálculo da entropia de *Shannon*

(complexidade da ordem de n), e por ser uma métrica comumente empregada em diversos trabalhos apresentados no capítulo anterior, esta técnica é utilizada nesta pesquisa para a detecção de ataques *LR DDoS slowloris*.

5 A PROPOSTA: ALGORITMO DE DETECÇÃO DE ATAQUES *SLOWLORIS*

Os sistemas de IDS não conseguem distinguir com eficiência ataques *slow* HTTP do tráfego normal uma vez que não são enviados pacotes mal formatados (HONG et al., 2017). O algoritmo proposto para a detecção de ataques *LR DDoS slowloris*, Detecta *Slowloris*, e sua análise são aqui descritos.

Em uma conexão HTTP normal, cada linha da mensagem termina com um caracter não-imprimível CRLF e, dois caracteres CRLF juntos identificam o fim da requisição HTTP. No ataque *slowloris* o atacante envia lentamente os caracteres de fim de cabeçalho para manter o servidor aguardando o cabeçalho completo e, assim esgotar os seus recursos. Uma boa métrica para a sua detecção é a taxa de recepção do marcador de fim de cabeçalho e a quantidade de pacotes com *payload* vazio por unidade de tempo.

Nesta pesquisa, foram criadas duas métricas denominadas PVS (Pacotes Vazios por Segundo) e FCS (Fim de Cabeçalho por Segundo). Sejam PVS (Pacotes Vazios por Segundo), a quantidade de pacotes com o tamanho do *payload* igual a zero recebidos por segundo e FCS (Fim de Cabeçalho por Segundo), a taxa de recepção do marcador de fim de cabeçalho e início da mensagem no protocolo HTTP (`\r\n\r\n`) por segundo.

Durante um ataque *slowloris*, ocorre um aumento no número de pacotes com *payload* vazio em função do envio de mensagens para manter a conexão ativa (*keep-alive*) e uma diminuição na taxa de envio do marcador de fim de cabeçalho do HTTP, conforme apresentado no algoritmo Detecta *Slowloris*.

Algorithm 1 Detecta *Slowloris* (Arquivo PCAP, PVS_SemAtaque, FCS_SemAtaque)

```
1: Slices[] ← Divide_slices_1min[ArquivoPCAP]
2: for I = 1 to Tamanho(Slices) do
3:     SliceAtual ← Slices[I]
4:     ArquivoIP[Or, Dst] ← Separa_IP_Origem_Destino[SliceAtual]
5:     EntropiaSliceAtual ← Calcula_entropia_IP_Or_Dst[ArquivoIP[Or, Dst]]
6:     PVS_SliceAtual ← PVS[SliceAtual]
7:     FCS_SliceAtual ← FCS[SliceAtual]
8:     if (EntropiaSliceAtual ≥ α * EntropiaSemAtaque) then
9:         if (PVS_SliceAtual ≥ β * PVS_SemAtaque) then
10:            if (FCS_SliceAtual ≤ γ * FCS_SemAtaque) then
11:                EmitirAlertaAtaque
12:            else
13:                EmitirAlertaSemAtaque
14:            end if
15:        end if
16:    end if
17: end for
```

Os coeficientes α, β e γ do algoritmo proposto assumem os valores 1.1, 1.6 e 0.6, respectivamente no *dataset* desenvolvido. No cálculo de α, β e γ foram utilizadas as médias nos slices de tempo de 1 minuto com um intervalo de confiança de 95%

O algoritmo proposto detectará o ataque *slowloris* comparando medidas históricas de entropia, PVS e FCS da rede com seus valores em um *slice* de tempo desejado, isto é, funciona de forma *offline* na rede. Como o custo computacional para o cálculo da entropia de *Shannon* é baixo (da ordem de n), é possível aplicar o algoritmo em tempo real, calculando os valores das métricas propostas e comparando-as. A função $Divide\_slices\_1min$ lê um arquivo PCAP, divide-o em *slices* de 1 minuto, devolvendo o $sliceAtual$. A função $Separa\_IP\_Origem\_Destino$, lerá o *slice*, buscará pelos *handshakes* entre o servidor *web* e os diversos clientes, gerando dois arquivos: um com IPs de origem e outro IPs de destino ($ArquivoIPs$).

A função $Calcula\_entropia\_IP\_Or\_Dst$ cuja entrada é $ArquivoIPs$, calcula a entropia de IP de origem (ou IP destino) utilizando a equação:

$$S = -\sum_{i=1}^{n} W \log_2(W) \tag{5.1}$$

Na Equação 5.1, ocorre uma média ponderada dos logaritmos das probabilidades, na qual os pesos são as probabilidades dos valores da variável aleatória W, indicando que a entropia S pode ser interpretada como o valor esperado da variável aleatória que assume o valor $log_2(W)$, com probabilidade W.

O valor de W é obtido pela contagem da frequencia de ocorrência dos endereços de IP em um determinado intervalo de tempo, sendo os dados normalizados (o cálculo da probabilidade) pela instrução Python $prob = counts/counts.sum()$.

A função PVS cuja entrada é o *sliceAtual*, retornará o número de pacotes com *payload* vazio por segundo ($PVS\_SliceAtual$), enquanto a função FCS devolverá o valor da taxa de recepção do marcador de fim de cabeçalho ($FCS\_SliceAtual$).

Será considerado ataque quando a entropia do *slice* atual ($entropiaSliceAtual$) exceder a entropia em um período sem ataques ($entropiaSemAtaq$) em pelo menos 10%, o PVS aumentar em pelo menos 60% e o FCS diminuir em pelo menos 40%.

Em função do tamanho dos arquivos PCAP gerados, tentar visualizá-los com *wireshark*, por exemplo, torna-se inviável devido ao consumo elevado de memória RAM. Um arquivo de 1 GB aberto no *wireshark*, consumia cerca de 6GB de memória RAM. A biblioteca *pyshark* (NEWT, 2017) também não se mostrou uma boa opção pelo mesmo motivo. A fim de cortar o arquivo em pedaços menores (*slices* de 1 min) e tornar mais eficiente o processamento, a ferramenta *opensource tracesplit*, que é incluída na *libtrace* foi utilizada (ALCOCK et al., 2012).

Para a presente dissertação, foram desenvolvidos *scripts* na linguagem Python para as seguintes ações:

- separar IP origem e IP destino em cada um arquivos fatiados pelo *tracesplit*;
- calcular as entropias de *Shannon* de acordo com a equação 4.2;
- plotar os gráficos (de entropia e outros) ao longo do tempo;e
- cálculo das métricas propostas para a detecção do ataque *slowloris* PVS e FCS.

Ao aplicar o algoritmo Detecta *Slowloris* no *dataset* desenvolvido nesta dissertação, obtem-se 120 *slices* com ataque *slowloris* e 180 *slices* sem ataques, totalizando 300 *slices* de 1 minuto cada.

A acurácia A de um algoritmo é definida como a proporção de classificações corretas (Verdadeiros Positivos, VP, e Verdadeiros Negativos,VN) para o total de elementos classificados (T):

$$A = \frac{VP + VN}{T} \tag{5.2}$$

A precisão P é definida como a razão entre o número de Verdadeiros Positivos (VP) e a soma dos Verdadeiros Positivos e Falsos Positivos (FP):

$$P = \frac{VP}{VP + FP} \tag{5.3}$$

O *Recall* R é a razão entre o número de Verdadeiros Positivos (VP) e a soma de Verdadeiros Positivos e Falsos Negativos (FN):

$$R = \frac{VP}{VP + FN} \tag{5.4}$$

A acurácia do algoritmo proposto, aplicado ao *dataset*, é de 91.7%, apresentando 1.7% de falsos positivos e 6.7% de falsos negativos, conforme a matriz de confusão apresentada na figura 5.1.

Não foi possível aplicar o algoritmo no *dataset* desenvolvido na pesquisa de (KUMAR et al., 2016a) que, apesar de conter o ataque *slowloris*, não disponibiliza os endereços de IP para o cálculo da entropia. Para a aplicação do algoritmo Detecta *Slowloris*, um *dataset* disponibilizado deve conter além dos endereços IPs para computar as entropias, os *payloads* dos pacotes a fim de calcular as métricas PVS e FCS .

O algoritmo proposto em (MA; CHEN, 2014) para a detecção de ataques *DDoS* utilizou a entropia de *Tsallis*, que é uma generalização da entropia de *Boltzmann-Gibbs* e o expoente de *Lyapunov*, com sensibilidade de 100% (nenhum falso negativo) e 100% de especificidade (0% de falsos positivos), mas não foi testado com ataques *LR DDoS*.

(MONGELLI et al., 2015) com a utilização de Transformada de *Fourier* e informação mútua, detectou ataques *LR DDoS* com sensibilidade de 85% e especificidade de 90%, enquanto (SAIED et al., 2016) com redes neurais artificiais atingiu uma acurácia de 98%.

Em sua aplicação de entropia para a detecção de ataques *DDoS* em um ambiente virtual na nuvem, o método de (CAO et al., 2015) apresentou 75% de falsos positivos quando o *throughput* da rede e o uso de CPU são menores que 30%. (ÖZÇELIK; BROOKS, 2015) demonstrou como a introdução de pacotes para aumentar os falsos positivos em um sistema que utiliza entropia na detecção de ataques *DDoS*, pode representar uma vulnerabilidade no mesmo (*spoofing* da entropia).

Verdadeiros Positivos 33.3 %	Falsos Negativos 6.7%	Acurácia 91.7%	Precisão 95.2%
Falsos Positivos 1.7%	Verdadeiros Negativos 58.3%	*Recall* 83.3%	

FIG. 5.1: Matriz de confusão do Algoritmo

6 ANÁLISE DOS RESULTADOS

Foi gerado um *dataset* com os ataques *slowloris* e *sockstress*, conforme a descrição do Apêndice desta Dissertação. Os experimentos foram realizados em um ambiente virtualizado com o *software* Vmware Workstation em 11 máquinas físicas, sendo 10 atacantes, cada uma com 7 máquinas virtuais Debian Linux versão 8.7 32 bits, e gerando tráfegos *slowloris* e *sockstress*. O tráfego de fundo foi gerado com a ferramenta httpmon, que gera tráfego HTTP com intervalo entre duas requisições consecutivas seguindo uma distribuição exponencial (GRIMALDI et al., 2015) e os ataques com a ferramenta slowhttptest (TRIPATHI et al., 2016). As amostras contendo o tráfego de rede coletado em laboratório com o tcpdump foram divididas em *time slices* de 1 minuto cada.

Pode-se inferir das figuras 6.1, 6.2, 6.3 e 6.4 que o desvio padrão do valor da entropia (de IP origem e IP destino) durante um período sem ataques é praticamente constante, enquanto que durante o ataque *slowloris* há uma grande variação deste desvio padrão.

A média da entropia do IP destino durante um período sem ataques foi de 4,30 $\pm 0,0043$, e a média da entropia do IP origem foi de 3,23 $\pm 0,0062$, conforme as figuras 6.1 e 6.3, respectivamente.

Durante um período de ataques *slowloris*, a média da entropia do IP destino foi de 2,65 $\pm 0,83$ e a média da entropia do IP origem foi de 3,68 $\pm 0,22$, de acordo com as figuras 6.2 e 6.4, respectivamente.

Utilizou-se o teste de Wilcoxon (Wilcoxon *Signed Rank Test*) (PRUNEAU, 2017) para a análise da variação de entropia das amostras antes e durante os ataques com significância estatística estabelecida em 5% ($p<0,05$) (confiança de 95 %).

As seguintes hipóteses foram testadas com intervalo de confiança de 95% :

- H_0: O ataque *slowloris* não tem efeito estatisticamente significante na variação da entropia das amostras;
- H_1: O ataque *slowloris* tem efeito estatisticamente significante na variação da entropia das amostras.

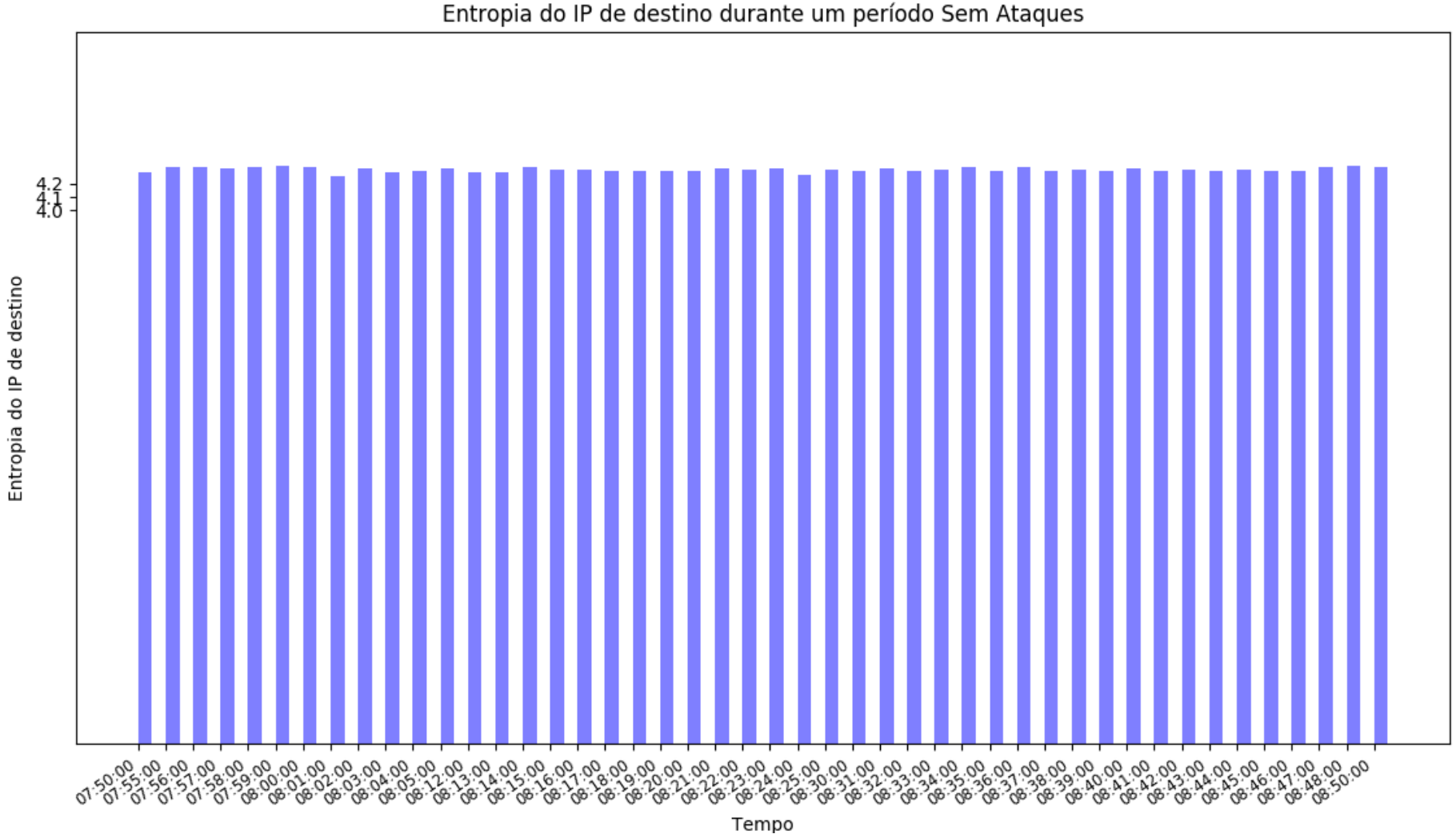

FIG. 6.1: Entropia do IP destino em um período sem ataques

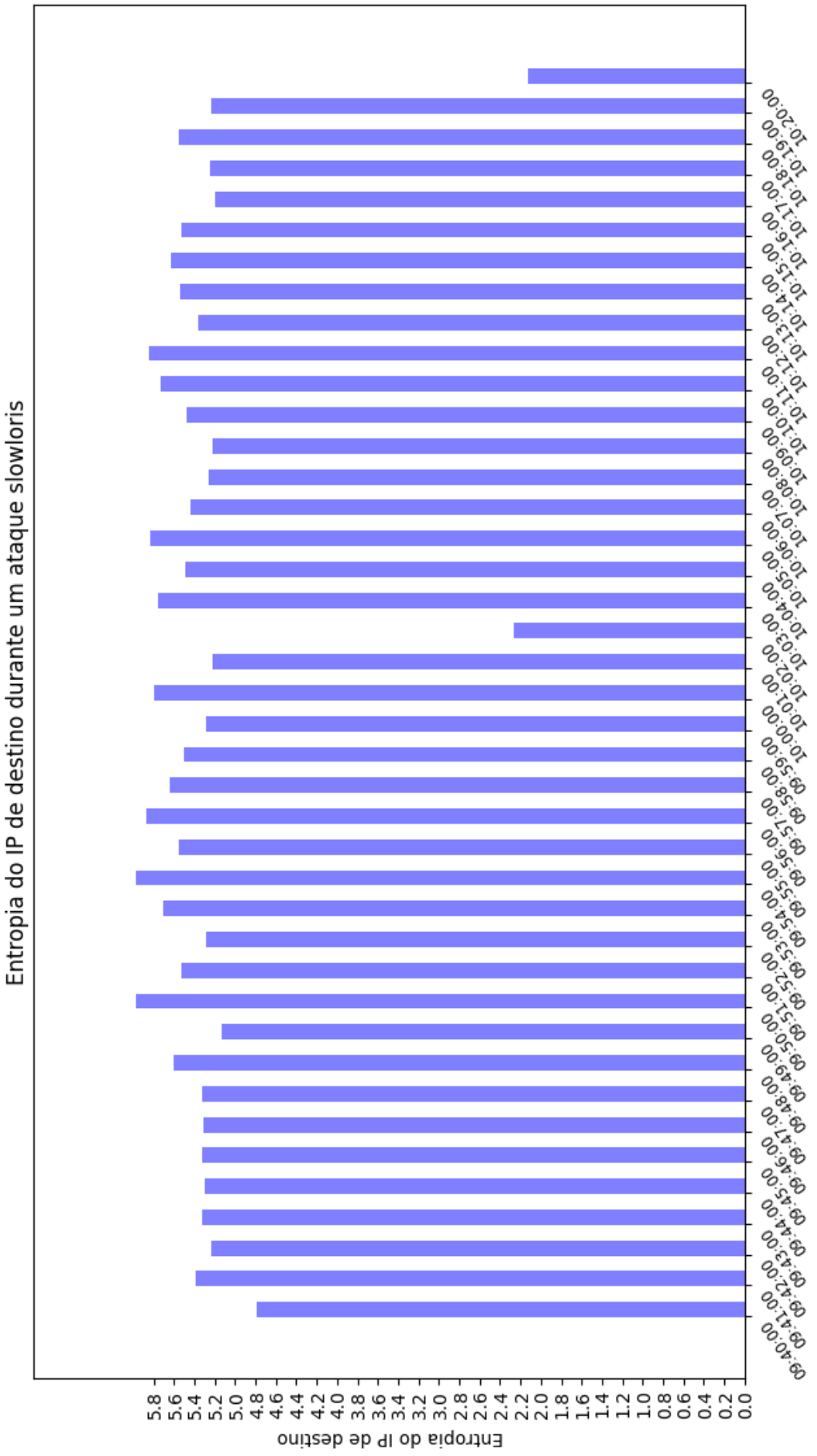

FIG. 6.2: Entropia do IP destino durante um ataque *slowloris*

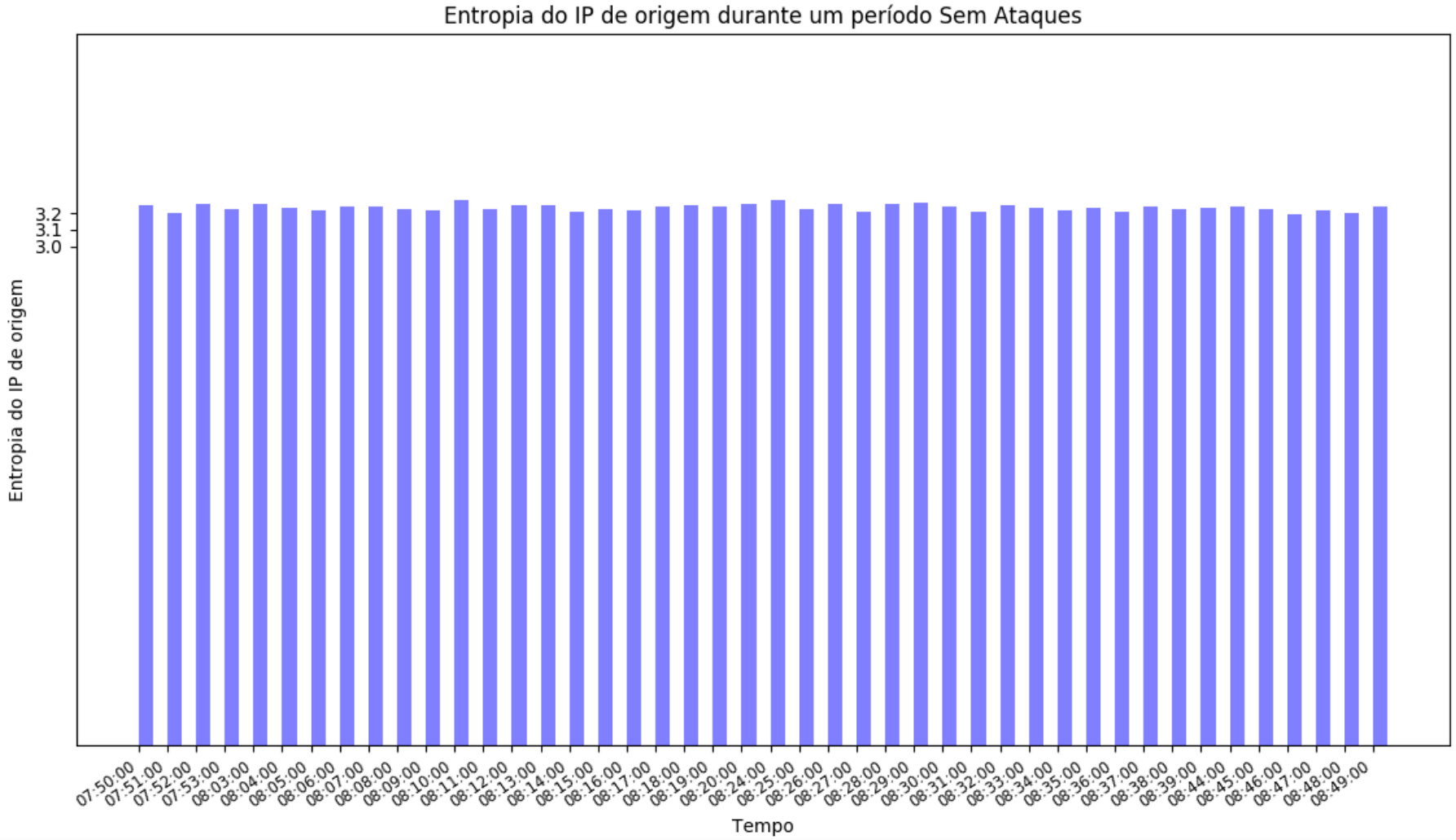

FIG. 6.3: Entropia do IP de origem em um período sem ataques

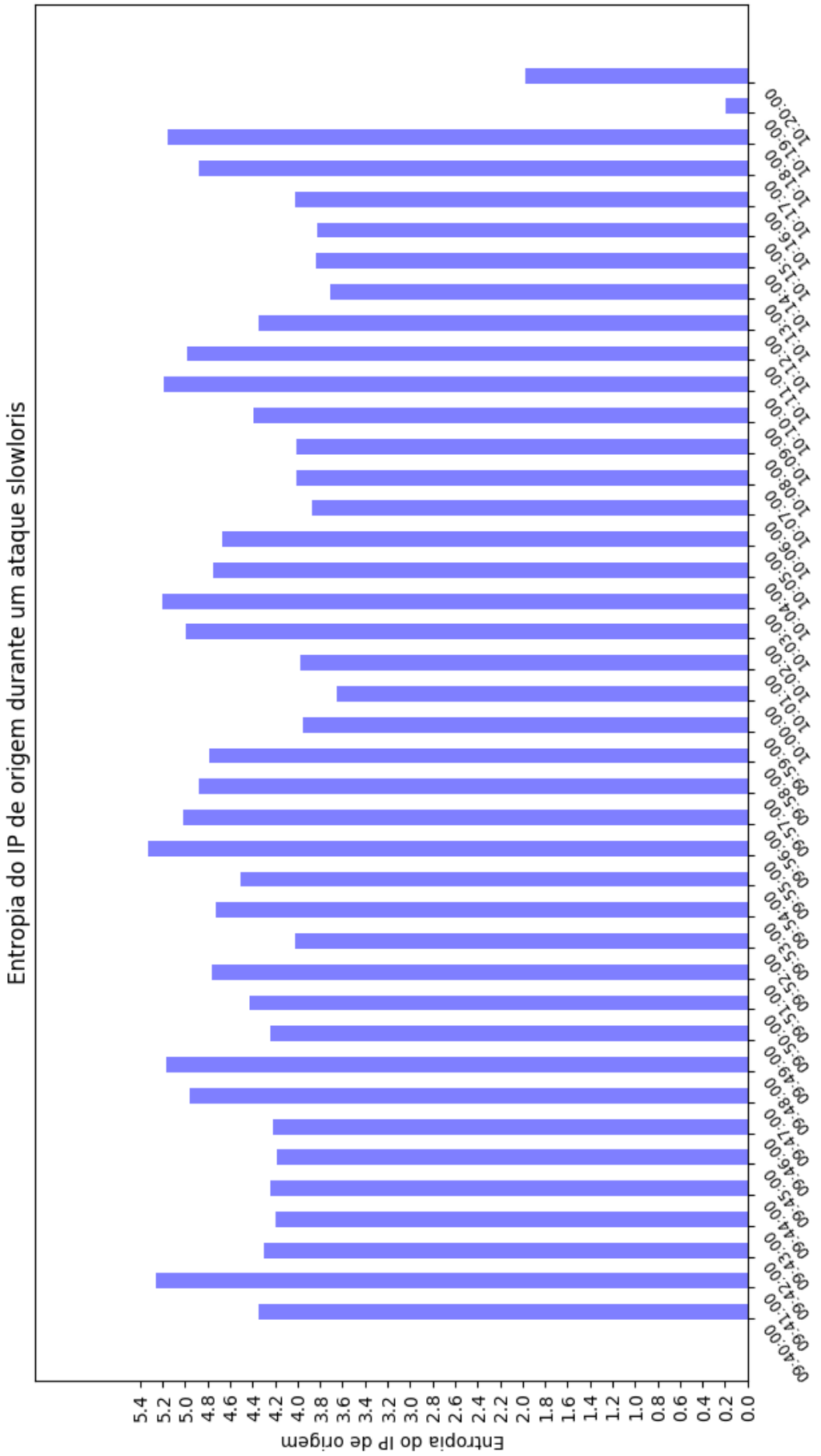

FIG. 6.4: Entropia do IP de origem durante um ataque *slowloris*

Há uma ligação estatisticamente significativa entre o aumento da entropia e o ataque *slowloris* com a aplicação do teste de Wilcoxon. Esses resultados são significativos no nível 0,05 (intervalo de confiança de 95 %).

7 CONSIDERAÇÕES FINAIS

Conforme os experimentos demonstraram, o ataque *slowloris* é efetivo em consumir os recursos de um servidor *web* Apache, especialmente a memória, e torná-lo indisponível. Este trabalho apresentou o novo *dataset* de ataques *LR DDoS* desenvolvido, que vem preencher uma lacuna na área, já que outros *datasets* não apresentam os ataques *slowloris* e *sockstress*.

Diferentemente do trabalho de (D'CRUZE et al., 2017) em que é criticada a utilização de medidas de entropia de IP em função da possibilidade de IP *spoofing*, como no ataque *slowloris* uma conexão TCP é estabelecida, os valores de entropia de IP origem e destino foram eficientes na detecção do ataque.

Analisando as figuras 6.1, 6.2, 6.3 e 6.4, infere-se que o desvio padrão do valor da entropia (de IP origem e IP destino) durante um período sem ataques é praticamente constante, enquanto que durante o ataque *slowloris* há uma grande variação do desvio padrão.

Encontrou-se, com o teste de Wilcoxon, uma ligação estatisticamente significativa entre o aumento da entropia, o aumento do PVS (Pacotes Vazios por Segundo), a diminuição da taxa de FCS (Fim de Cabeçalho por Segundo) e o ataque *slowloris*, com intervalo de confiança de 95%).

Em função da característica do ataque *slowloris* de explorar o funcionamento normal do protocolo HTTP de aguardar a recepção do duplo CRLF, duas métricas para a detecção do ataque *slow* HTTP *header* são PVS (Pacotes Vazios por Segundo), a quantidade de pacotes com o tamanho do *payload* igual a zero por segundo e FCS (Fim de Cabeçalho por Segundo), a taxa de recepção do marcador de fim de cabeçalho e início da mensagem no protocolo HTTP (`\r\n\r\n`) por segundo.

O algoritmo proposto apresenta uma acurácia de 91.7% e uma precisão de 95.2%, apresentando 1.7% de falsos positivos e 6.7% de falsos negativos.

Utilizando 70 máquinas virtuais atacantes, foram geradas em média 72 conexões por segundo durante o ataque *slowloris* e o servidor Apache teve um incremento de 8% na utilização de CPU, 27,3% em memoria, 10,8 vezes em pacotes recebidos e 8,3 vezes em pacotes enviados.

É importante destacar que não apenas servidores Apache são vulneráveis aos ataques *slow* HTTP, mas também servidores Microsoft IIS (TRIPATHI; HUBBALLI, 2018).

É muito difícil para um sistema de IDS distinguir um ataque do tipo *slow* HTTP do tráfego normal de um servidor, já que não são geradas requisições mal formatadas.

7.1 TRABALHOS FUTUROS

Durante o desenvolvimento desta dissertação, foram identificadas diversas possibilidades de trabalhos futuros. Na simulação dos ataques, por exemplo, optou-se pela utilização de máquinas virtuais em laboratório e na geração de tráfego de fundo com o httpmon, que gera tráfego com distribuição exponencial. O *NS-3: Network Simulator-3* (RILEY; HENDERSON, 2010), que é um simulador de redes de camadas de enlace, redes e transporte com grande escalabilidade e que usa as linguagens de programação C++ ou Python pode ser testado, permitindo, por exemplo, a variação de topologias de rede. Devido ao elevado número de equipamentos *IoT* conectados à internet, pode-se testar uma topologia em que estes dispositivos sejam a origem e/ou o destino de ataques *LR DDoS*.

Uma das limitações nos testes do algoritmo proposto Detecta *Slowloris*, é a ausência de outros *datasets* publicamente disponíveis contendo o ataque *slowloris* e sem ocultar os endereços de IP e os *payloads* dos pacotes. Assim, seria desejável, aplicar o algoritmo em futuros *datasets* disponíveis, comparando-o com outros algoritmos existentes. Pode-se também, conforme as pesquisas de (ÖZÇELIK; BROOKS, 2015), verificar o impacto na detecção do ataque *slowloris* da injeção de pacotes para realizar o *spoofing* da entropia, testando, além disso, outras métricas de entropia como a entropia de *Kullback–Leibler*.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ABNT, A. D. N. Iec 27.002: 2005 (antiga nbr iso/iec 17799: 2005)-código de prática para a gestão da segurança da informação. **Rio do Janeiro: ABNT**, v. 1, p. 4, 2013.

ADI, E.; BAIG, Z. A.; HINGSTON, P. ; LAM, C.-P. Distributed denial-of-service attacks against http/2 services. **Cluster Computing**, v. 19, n. 1, p. 79–86, 2016.

ALCOCK, S.; LORIER, P. ; NELSON, R. Libtrace: a packet capture and analysis library. **ACM SIGCOMM Computer Communication Review**, v. 42, n. 2, p. 42–48, 2012.

ALHARBI, T.; ALJUHANI, A.; LIU, H. ; HU, C. Smart and lightweight ddos detection using nfv. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON COMPUTE AND DATA ANALYSIS, 1., 2017. **Anais**... [S.l.: s.n.], 2017, p. 220–227.

ALOMARI, E.; MANICKAM, S.; GUPTA, B.; KARUPPAYAH, S. ; ALFARIS, R. Botnet-based distributed denial of service (ddos) attacks on web servers: classification and art. **arXiv preprint arXiv:1208.0403**, v. 2012, p. 2–8, 2012.

ASWATHY, R.; MALARVIZHI, N. Internet of things (iot): a survey on protocols and security risks. **International Journal of Engineering & Technology**, v. 7, n. 1.7, p. 15–20, 2018.

BANERJEE, A. Noc: An introduction to information theory. **Technology & Applied Sciences**, v. 1, 2016. Disponível em: <http://doer.col.org/handle/123456789/6536>. Acesso em: 2017-09-07.

BARATI, M.; ABDULLAH, A.; UDZIR, N. I.; MAHMOD, R. ; MUSTAPHA, N. Distributed denial of service detection using hybrid machine learning technique. In: BIOMETRICS AND SECURITY TECHNOLOGIES (ISBAST), 2014 INTERNATIONAL SYMPOSIUM ON, 2014., 2014. **Anais**... [S.l.: s.n.], 2014, p. 268–273.

BECKETT, D.; SEZER, S. ; MCCANNY, J. New sensing technique for detecting application layer ddos attacks targeting back-end database resources. In: COMMUNICATIONS (ICC), 2017 IEEE INTERNATIONAL CONFERENCE ON, 1., 2017. **Anais**... [S.l.: s.n.], 2017, p. 1–7.

BEHAL, S.; KUMAR, K. Trends in validation of ddos research. **Procedia Computer Science**, v. 85, p. 7–15, 2016.

BELSHE, M.; PEON, R. ; THOMSON, M. Rfc 7540: hypertext transfer protocol version 2 (http/2). **Internet Engineering Task Force (IETF)//BitGo, Google Inc.//May**, v. 2, p. 1–4, 2015.

BERTINO, E.; ISLAM, N. Botnets and internet of things security. **Computer**, v. 50, n. 2, p. 76–79, 2017.

BHATTACHARYYA, D. K.; KALITA, J. K. **DDoS attacks**. [S.l.]: CRC Press, 2016.

BHUYAN, M. H.; BHATTACHARYYA, D. ; KALITA, J. K. An empirical evaluation of information metrics for low-rate and high-rate ddos attack detection. **Pattern Recognition Letters**, v. 51, p. 1–7, 2015.

BOTEANU, D.; FERNANDEZ, J. M. A comprehensive study of queue management as a dos counter-measure. **International journal of information security**, v. 12, n. 5, p. 347–382, 2013.

BROOKS, R. R.; OTHERS. Cusum-entropy: an efficient method for ddos attack detection. In: 2016 4TH INTERNATIONAL ISTANBUL SMART GRID CONGRESS AND FAIR (ICSG), 1., 2016. **Anais**... [S.l.: s.n.], 2016, p. 1–5.

CAO, J.; YU, B.; DONG, F.; ZHU, X. ; XU, S. Entropy-based denial-of-service attack detection in cloud data center. **Concurrency and Computation: Practice and Experience**, v. 27, n. 18, p. 5623–5639, 2015.

CARL, G.; KESIDIS, G.; BROOKS, R. R. ; RAI, S. Denial-of-service attack-detection techniques. **IEEE Internet Computing**, v. 10, n. 1, p. 82–89, 2006.

NATIONAL CYBER SECURITY CENTRE. TCP sockstress: Several Vulnerabilities in the Implementation of TCP. Disponível em: <https://www.ncsc.nl/english/current-topics/factsheets/factsheet-tcp-sockstress.html>. Acesso em: 2016-11-03.

COVER, T. M.; THOMAS, J. A. **Elements of information theory**. [S.l.]: John Wiley & Sons, 2012.

CRISCUOLO, P. J. **Distributed denial of service: Trin00, tribe flood network, tribe flood network 2000, and stacheldraht ciac-2319**. [S.l.: s.n.], 2000. (Relatório Técnico).

CROSS, K. 4 steps to defeat a ddos attack on your organisation. **Database and Network Journal**, v. 41, n. 5, p. 16, 2011.

CUSACK, B.; LUTUI, R. ; KHALEGHPARAST, R. Detecting slow ddos attacks on mobile devices. **Australasian Conference on Information Systems**, v. 1, 2016. Disponível em: <http://ro.uow.edu.au/acis2016/papers/1/57/>. Acesso em: 2017-09-22.

DAVID, J.; THOMAS, C. Ddos attack detection using fast entropy approach on flow-based network traffic. **Procedia Computer Science**, v. 50, p. 30–36, 2015.

DIMITROVA, B.; MILEVA, A. Steganography of hypertext transfer protocol version 2 (http/2). **Journal of Computer and Communications**, v. 5, p. 98–111, 2017.

DITTRICH, D.; MIRKOVIC, J.; REIHER, P. ; DIETRICH, S. **Internet Denial of Service: Attack and Defense Mechanisms**. [S.l.]: Pearson Education, 2004.

DURAVKIN, I.; LOKTIONOVA, A. ; CARLSSON, A. Method of slow-attack detection. In: PROBLEMS OF INFOCOMMUNICATIONS SCIENCE AND TECHNOLOGY, 2014 FIRST INTERNATIONAL SCIENTIFIC-PRACTICAL CONFERENCE, 1., 2014. **Anais**... [S.l.: s.n.], 2014, p. 171–172.

D'CRUZE, H.; WANG, P.; SBEIT, R. O. ; RAY, A. A software-defined networking (sdn) approach to mitigating ddos attacks. **Advances in Intelligent Systems and Computing**, v. 558, p. 141–145, 2017.

EPSTEIN, I. **Teoria da informação**. [S.l.]: Editora Atica, 1986.

FEINSTEIN, L.; SCHNACKENBERG, D.; BALUPARI, R. ; KINDRED, D. Statistical approaches to ddos attack detection and response. In: DARPA INFORMATION SURVIVABILITY CONFERENCE AND EXPOSITION, 2003. PROCEEDINGS, 2003., 2003. **Anais**... [S.l.: s.n.], 2003, p. 303–314.

GAVRILIS, D.; DERMATAS, E. Real-time detection of distributed denial-of-service attacks using rbf networks and statistical features. **Computer Networks**, v. 48, n. 2, p. 235–245, 2005.

O GLOBO. Ataque hacker derruba parte da internet nos EUA. Disponível em: <http://oglobo.globo.com/sociedade/tecnologia/ataque-hacker-derruba-parte-da-internet-nos-eua-20332302>. Acesso em: 2016-10-21.

ROBERT GRAHAM. Slowloris all the things. Disponível em: <http://blog.erratasec.com/2017/07/slowloris-all-things.html>. Acesso em: 2017-07-26.

GRIMALDI, D.; PERSICO, V.; PESCAPÉ, A.; SALVI, A. ; SANTINI, S. A feedback-control approach for resource management in public clouds. In: GLOBAL COMMUNICATIONS CONFERENCE (GLOBECOM), 2015 IEEE, 2015., 2015. **Anais**... [S.l.: s.n.], 2015, p. 1–7.

HAMDI, M.; BOUDRIGA, N. Detecting denial-of-service attacks using the wavelet transform. **Computer Communications**, v. 30, n. 16, p. 3203–3213, 2007.

HAVOC. Sockstress. Disponível em: <http://www.mit.edu/afs.new/sipb/user/tboning/dos-tools/sockstress/README.txt>. Acesso em: 2016-11-03.

HONG, K.; KIM, Y.; CHOI, H. ; PARK, J. Sdn-assisted slow http ddos attack defense method. **IEEE Communications Letters**, v. 37, 2017. Disponível em: <https://publikationen.uni-tuebingen.de/xmlui/bitstream/handle/10900/78143/KuVS-FG-Netsoft17-3.pdf?sequence=1>. Acesso em: 2017-11-07.

HOQUE, N.; KASHYAP, H. ; BHATTACHARYYA, D. Real-time ddos attack detection using fpga. **Computer Communications**, v. 110, p. 48–58, 2017.

HOQUE, N.; BHATTACHARYYA, D. K. ; KALITA, J. K. A novel measure for low-rate and high-rate ddos attack detection using multivariate data analysis. In: 2016 8TH INTERNATIONAL CONFERENCE ON COMMUNICATION SYSTEMS AND NETWORKS (COMSNETS), 2016., 2016. **Anais**... [S.l.: s.n.], 2016, p. 1–2.

TAYLOR HORNBY. Sockstress. Disponível em: <https://defuse.ca/sockstress.htm>. Acesso em: 2016-10-28.

TAYLOR HORNBY. Sockstress Tool. Disponível em: <https://github.com/defuse/sockstress>. Acesso em: 2016-11-02.

JAZI, H. H.; GONZALEZ, H.; STAKHANOVA, N. ; GHORBANI, A. A. Detecting http-based application layer dos attacks on web servers in the presence of sampling. **Computer Networks**, v. 121, p. 25–36, 2017.

JIAN-QI, Z.; FENG, F.; KE-XIN, Y. ; YAN-HENG, L. Dynamic entropy based dos attack detection method. **Computers & Electrical Engineering**, v. 39, n. 7, p. 2243–2251, 2013.

JUNGMANN, R. Estrategia nacional de defesa (end). **JOBIM, Nelson A**, v. 1, 2012. Disponível em: <http://www.defesa.gov.br/arquivos/2012/mes07/end.pdf>. Acesso em: 2016-12-02.

KASPERSKY. Kaspersky DDoS Intelligence Report for Q1 2016. Disponível em: <https://securelist.com/analysis/quarterly-malware-reports/74550/kaspersky-ddos-intelligence-report-for-q1-2016/>. Acesso em: 2016-04-28.

KATKAR, V.; ZINJADE, A.; DALVI, S.; BAFNA, T. ; MAHAJAN, R. Detection of dos/ddos attack against http servers using naive bayesian. In: COMPUTING COMMUNICATION CONTROL AND AUTOMATION (ICCUBEA), 2015 INTERNATIONAL CONFERENCE ON, 1., 2015. **Anais**... [S.l.: s.n.], 2015, p. 280–285.

KATKAR, V. D.; BHATIA, D. S. Experiments on detection of denial of service attacks using reptree. In: GREEN COMPUTING, COMMUNICATION AND CONSERVATION OF ENERGY (ICGCE), 2013 INTERNATIONAL CONFERENCE ON, 2013., 2013. **Anais**... [S.l.: s.n.], 2013, p. 713–718.

KHARAT, J.; NAIK, R. A vivacious approach to detect and prevent ddos attack. **International Journal of Research in Engineering and Technology**, v. 2, n. 10, p. 434–440, 2013.

KSHIRSAGAR, D.; SAWANT, S.; RATHOD, A. ; WATHORE, S. Cpu load analysis & minimization for tcp syn flood detection. **Procedia Computer Science**, v. 85, p. 626–633, 2016.

KUMAR, R.; LAL, S. P. ; SHARMA, A. Detecting denial of service attacks in the cloud. In: DEPENDABLE, AUTONOMIC AND SECURE COMPUTING, 14TH INTL CONF ON PERVASIVE INTELLIGENCE AND COMPUTING, 2016 IEEE 14TH INTL C, 2016., 2016. **Anais**... [S.l.: s.n.], 2016, p. 309–316.

KUMAR, V.; KUMAR, K. ; OTHERS. Classification of ddos attack tools and its handling techniques and strategy at application layer. In: ADVANCES IN COMPUTING, COMMUNICATION, & AUTOMATION (ICACCA)(FALL), INTERNATIONAL CONFERENCE ON, 2016., 2016. **Anais**... [S.l.: s.n.], 2016, p. 1–6.

KUZMANOVIC, A.; KNIGHTLY, E. W. Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants. In: PROCEEDINGS OF THE 2003 CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATIONS, 2003., 2003. **Anais**... [S.l.: s.n.], 2003, p. 75–86.

KUZMANOVIC, A.; KNIGHTLY, E. W. Low-rate tcp-targeted denial of service attacks and counter strategies. **IEEE/ACM Transactions on Networking (TON)**, v. 14, n. 4, p. 683–696, 2006.

LUO, J.; YANG, X.; WANG, J.; XU, J.; SUN, J. ; LONG, K. On a mathematical model for low-rate shrew ddos. **IEEE Transactions on information Forensics and Security**, v. 9, n. 7, p. 1069–1083, 2014.

MA, X.; CHEN, Y. Ddos detection method based on chaos analysis of network traffic entropy. **IEEE Communications Letters**, v. 18, n. 1, p. 114–117, 2014.

MANDARINO JUNIOR, R.; CANONGIA, C. Livro verde: segurança cibernética no brasil. **Brasília: GSIPR/SE/DSIC**, v. 1, p. 5–15, 2010.

MANSFIELD-DEVINE, S. Anonymous: serious threat or mere annoyance?. **Network Security**, v. 2011, n. 1, p. 4–10, 2011.

DAVE MCMILLEN. Dissecting a Hacktivist DDoS Tool: Saphyra Revealed. Disponível em: <https://securityintelligence.com/dissecting-hacktivists-ddos-tool-saphyra-revealed/>. Acesso em: 2016-11-05.

MIAO, R.; YU, M. ; JAIN, N. Nimbus: cloud-scale attack detection and mitigation. **ACM SIGCOMM Computer Communication Review**, v. 44, n. 4, p. 121–122, 2015.

MIRKOVIC, J.; REIHER, P. A taxonomy of ddos attack and ddos defense mechanisms. **ACM SIGCOMM Computer Communication Review**, v. 34, n. 2, p. 39–53, 2004.

MONGELLI, M.; AIELLO, M.; CAMBIASO, E. ; PAPALEO, G. Detection of dos attacks through fourier transform and mutual information. In: COMMUNICATIONS (ICC), 2015 IEEE INTERNATIONAL CONFERENCE ON, 1., 2015. **Anais**... [S.l.: s.n.], 2015, p. 7204–7209.

P. NARINDER. Lloyds Banking Group hit with distributed denial of service attack. Disponível em: <http://www.welivesecurity.com/2017/01/24/lloyds-banking-group-hit-distributed-denial-service-attack/>. Acesso em: 2017-01-24.

KIMI NEWT. Pyshark. Disponível em: <https://pypi.python.org/pypi/pyshark>. Acesso em: 2017-02-24.

ÖZÇELIK, İ.; BROOKS, R. R. Deceiving entropy based dos detection. **Computers & Security**, v. 48, p. 234–245, 2015.

JONATHAN ARTHUR PAES. LOIC - Ataque DDoS com o android. Disponível em: <http://www.androidhyperuser.com/2016/04/ataque-ddos-com-o-android.html>. Acesso em: 2016-11-05.

PARK, J.; IWAI, K.; TANAKA, H. ; KUROKAWA, T. Analysis of slow read dos attack and countermeasures on web servers. **International Journal of Cyber-Security and Digital Forensics (IJCSDF)**, v. 4, n. 2, p. 339–353, 2015.

PHARANDE, S.; PAWAR, P.; WANI, P. ; PATKI, A. Application of hurst parameter and fuzzy logic for denial of service attack detection. In: ADVANCE COMPUTING CONFERENCE (IACC), 2015 IEEE INTERNATIONAL, 1., 2015. **Anais**... [S.l.: s.n.], 2015, p. 834–838.

PRUNEAU, C. **Data Analysis Techniques for Physical Scientists**. [S.l.]: Cambridge University Press, 2017.

RADWARE. **DDoS Handbook: The Ultimate Guide to Everything You Need to Know about DDoS Attacks**. [S.l.]: Radware, 2016.

RADWARE. Seven Common DDoS Attack Tools. Disponível em: <https://security.radware.com/ddos-knowledge-center/ddos-attack-types/common-ddos-attack-tools/>. Acesso em: 2016-11-05.

RAI, A.; CHALLA, R. K. Survey on recent ddos mitigation techniques and comparative analysis. In: COMPUTATIONAL INTELLIGENCE & COMMUNICATION TECHNOLOGY (CICT), 2016 SECOND INTERNATIONAL CONFERENCE ON, 1., 2016. **Proceedings**... [S.l.: s.n.], 2016, p. 96–101.

ALISON DENISCO RAYOME. DDoS attacks increased 91<https://www.techrepublic.com/article/ddos-attacks-increased-91-in-2017-thanks-to-iot/>. Acesso em: 2017-11-20.

RILEY, G. F.; HENDERSON, T. R. The ns-3 network simulator. **International Journal of Engineering & Technology**, v. 238, p. 15–34, 2010.

RMAYTI, M.; BEGRICHE, Y.; KHATOUN, R.; KHOUKHI, L. ; GAITI, D. Denial of service (dos) attacks detection in manets through statistical models. In: 2014 GLOBAL INFORMATION INFRASTRUCTURE AND NETWORKING SYMPOSIUM (GIIS), 2014., 2014. **Anais**... [S.l.: s.n.], 2014, p. 1–3.

SAIED, A.; OVERILL, R. E. ; RADZIK, T. Detection of known and unknown ddos attacks using artificial neural networks. **Neurocomputing**, v. 172, p. 385–393, 2016.

SANTOS, A. F. P.; SILVA, R. S. ; DE JANEIRO-RJ-BRASIL, R. Detecção de ataques ddos com gráficos de controle e bases de regras nebulosas. **International Information and Telecommunication and Technologies Conference**, v. 13, p. 101–118, 2010.

SELVARAJ, R.; KUTHADI, V. M. ; MARWALA, T. Ant-based distributed denial of service detection technique using roaming virtual honeypots. **IET Communications**, v. 10, n. 8, p. 929–935, 2016.

SHAFIEIAN, S.; ZULKERNINE, M. ; HAQUE, A. Cloudzombie: Launching and detecting slow-read distributed denial of service attacks from the cloud. In: COMPUTER AND INFORMATION TECHNOLOGY; UBIQUITOUS COMPUTING AND COMMUNICATIONS; DEPENDABLE, AUTONOMIC AND SECURE COMPUTING; PERVASIVE INTELLIGENCE AND COMPUTING (CIT/IUCC/DASC/PICOM), 2015 IEEE INTERNATIONAL CONFERENCE ON, 2., 2015. **Anais**... [S.l.: s.n.], 2015, p. 1733–1740.

SHAH, V. M.; AGARWAL, A. Reliable alert fusion of multiple intrusion detection systems.. **IJ Network Security**, v. 19, n. 2, p. 182–192, 2017.

SHANNON, C. E.; WEAVER, W. **The mathematical theory of communication**. [S.l.]: University of Illinois press, 1998.

SHARMA, R. Detection of low rate dos attacks against http servers using spectral analysis. **School of Computer Science and Communication**, v. 1, p. 22–33, 2014.

SHARMA, S.; SAHU, S. K. ; JENA, S. K. On selection of attributes for entropy based detection of ddos. In: ADVANCES IN COMPUTING, COMMUNICATIONS AND INFORMATICS (ICACCI), 2015 INTERNATIONAL CONFERENCE ON, 2015., 2015. **Anais**... [S.l.: s.n.], 2015, p. 1096–1100.

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

SILVA, N. R.; SALLES, R. M. Métricas para a deteccão de ataques ddos. **Revista Militar de Ciência e Tecnologia**, v. 32, n. 2, p. 27–48, 2015.

SILVA, S. S.; SILVA, R. M.; PINTO, R. C. ; SALLES, R. M. Botnets: A survey. **Computer Networks**, v. 57, n. 2, p. 378–403, 2013.

SOCPEDIA. Biggest British Hosting Company 123-Reg Suffers Major DDoS Attack. Disponível em: <https://www.socpedia.com/biggest-british-hosting-company-123-reg-suffers-major-ddos-attack>. Acesso em: 2017-01-10.

STALLING, W. **Network Security Essentials**. [S.l.]: Pearson Education, 2016.

TAN, Z.; JAMDAGNI, A.; HE, X.; NANDA, P.; LIU, R. P. ; HU, J. Detection of denial-of-service attacks based on computer vision techniques. **IEEE transactions on computers**, v. 64, n. 9, p. 2519–2533, 2015.

TANENBAUM, A. S. **Redes de computadores**. [S.l.]: Pearson Education, 2011.

TAYAMA, S.; TANAKA, H. Analysis of slow read dos attack and communication environment. In: INTERNATIONAL CONFERENCE ON MOBILE AND WIRELESS TECHNOLOGY, 2017., 2017. **Anais**... [S.l.: s.n.], 2017, p. 350–359.

THOMSON, G.; CONRAD, J. Anonymous-arrests, leaks and infections. **NETWORK SECURITY**, v. 1, n. 1, p. 2–2, 2011.

TRIPATHI, N.; HUBBALLI, N. Slow rate denial of service attacks against http/2 and detection. **Computers & Security**, v. 72, p. 255–272, 2017.

TRIPATHI, N.; HUBBALLI, N. Slow rate denial of service attacks against http/2 and detection. **Computers & Security**, v. 72, p. 255–272, 2018.

TRIPATHI, N.; HUBBALLI, N. ; SINGH, Y. How secure are web servers? an empirical study of slow http dos attacks and detection. In: AVAILABILITY, RELIABILITY AND SECURITY (ARES), 2016 11TH INTERNATIONAL CONFERENCE ON, 2016., 2016. **Anais**... [S.l.: s.n.], 2016, p. 454–463.

VINÍCIUS VIEIRA. Conheça uma ferramenta excelente para ataques DDoS. Disponível em: <http://sejalivre.org/conheca-uma-ferramenta-excelenta-para-ataques-ddos/>. Acesso em: 2016-11-05.

VUONG, T. P.; LOUKAS, G.; GAN, D. ; BEZEMSKIJ, A. Decision tree-based detection of denial of service and command injection attacks on robotic vehicles. In: INFORMATION FORENSICS AND SECURITY (WIFS), 2015 IEEE INTERNATIONAL WORKSHOP ON, 2015., 2015. **Anais**... [S.l.: s.n.], 2015, p. 1–6.

YU, S. **Distributed Denial of Service Attack and Defense**. [S.l.]: Springer, 2014.

YU, S.; ZHOU, W.; DOSS, R. ; JIA, W. Traceback of ddos attacks using entropy variations. **IEEE Transactions on Parallel and Distributed Systems**, v. 22, n. 3, p. 412–425, 2011.

ZARGAR, S. T.; JOSHI, J. ; TIPPER, D. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. **IEEE Communications Surveys & Tutorials**, v. 15, n. 4, p. 2046–2069, 2013.

ZLOMISLIĆ, V.; FERTALJ, K. ; SRUK, V. Denial of service attacks, defences and research challenges. **Cluster Computing**, v. 2017, p. 1–11, 2017.

9 APÊNDICES

APÊNDICE 1: OS PRINCIPAIS *DATASETS* PARA AS PESQUISAS DE ATAQUES *DDOS*

Há poucos *datasets* disponíveis publicamente para a pesquisa de ataques *DDoS* na camada de aplicação (JAZI et al., 2017); o *dataset* originado desta pesquisa será disponibilizado.

Em relação a validação de experimentos com *datasets*, apenas o CAIDA *DDoS Attack* 2007, o DEFCON-10 *dataset*, o KDD *Cup* 1999 *dataset*, o TUIDS *DDoS dataset*, o DARPA *Intrusion Detection dataset*, e o NSL-KDD *dataset* são extensivamente usados, ao passo que datasets MIT Lincoln's LLSDOS 1.0 e 2.0 está obsoleto (BEHAL; KUMAR, 2016).

No CAIDA *DDoS Attack* 2007 *dataset*, o *payload* dos pacotes foi totalmente removido e os cabeçalhos estão disponíveis até a camada de transporte (SHIRAVI et al., 2012).

O DEFCON-10 *dataset* foi criado em 2002. O seu tráfego foi obtido na competição *Capture The Flag* (CTF), sendo, pois, muito diferente de um tráfego realístico (contém apenas conexões maliciosas como *port scans* e varreduras) sem um tráfego de fundo normal; é muito utilizado para a avaliação de IDS (SHIRAVI et al., 2012).

O *dataset Knowledge Discovery and Data Mining* (KDD Cup 1999) é muito utilizado em pesquisas de *malwares* mas não é adequado para a avaliação de detecção de ataques *DDoS* ou *flash events*. Alguns problemas do *dataset* são a grande quantidade de dados redundantes, em torno de 75%, dados de ataques falsos e antiquados para avaliação (BEHAL; KUMAR, 2016). O *dataset* KDD *Cup* 1999 não contém os tipos de ataques mais recentes. O TUIDS DDoS *dataset* foi desenvolvido no final de 2015, no entanto, muitos ataques estão desatualizados.

O DARPA *Intrusion Detection dataset*, sendo o DARPA *Intrusion Detection dataset* 2000 o mais utilizado, não abrange novas técnicas de ataque. Contém registros de auditoria do sistema operacional Windows NT e ataques *DoS* como o *neptune*, o *smurf* e o *ping of death*. É ainda útil para testes de IDS (SHAH; AGARWAL, 2017).

O NSL-KDD *dataset* foi desenvolvido para resolver alguns problemas inerentes ao KDD *Cup* 1999 *dataset*. Assim, sendo, não contém os dados reduntantes. A comparação entre os *datasets* está resumida na tabela 9.1.

Existe, ainda, o *dataset* desenvolvido na pesquisa de (KUMAR et al., 2016a) que, apesar de conter o ataque *slowloris*, não disponibiliza dados suficientes para o cálculo da entropia.

Há outras limitações no uso de *datasets*, como os supracitados, na validação de pes-

TAB. 9.1: Comparação entre os *datasets*

Dataset	Ataque *Slowloris*	Observações
CAIDA *DDoS*	Não	Sem *payload*. Algumas *flags* e informações de protocolos removidas.
DARPA 2000	Não	Arquivo de captura completo.
KDD-99	Não	Tráfego de fundo e ataques não estão bem identificados.
TUIDS *DDoS*	Não	Arquivo de captura completo.
DEFCON-10	Não	Apenas tráfego intrusivo.
NSL-KDD	Não	Arquivo de captura completo.

quisas em *DDoS* (BEHAL; KUMAR, 2016):

- ataques *LR* que são difíceis de detectar não são capturados;
- o endereço IP é diferente do verdadeiro;
- datasets de *flash events* são obsoletos;e
- muitos dos *datasets* disponíveis ocultam os detalhes específicos da aplicação.

APÊNDICE 2: FERRAMENTAS PARA ATAQUES *DDOS*

Existem diversas ferramentas para efetuar ataques de negação de serviço, sendo as suas características mais importantes resumidas na tabela 9.2 e apresentadas abaixo:

a) Slowhttptest: possui diversos módulos que permitem gerar os tipos de ataques *Slow* HTTP como *slowloris*, *slow read* e *slow body* (TRIPATHI et al., 2016).

b) *LOIC*: gera uma massiva quantidade de requisições HTTP, TCP e UDP. Pode ser usada como uma *botnet* controlada por IRC(Internet Relay Chat). (PAES, 2016), (MANSFIELD-DEVINE, 2011), (ZLOMISLIĆ et al., 2017);

c) *WebLOIC*: é a versão *Javascript* da ferramenta *LOIC* para o sistema operacional *Android* e que contém apenas a funcionalidade de HTTP *flooding* (ZLOMISLIĆ et al., 2017);

d) *THC-SSL-DoS*: criada pelo grupo alemão *The Hackers Choice*: THC e explora a assimetria do protocolo SSL (estabelecer uma conexão segura SSL requer um processamento muito maior no servidor que no cliente) (VIEIRA, 2011), (ZLOMISLIĆ et al., 2017);

e) *HOIC*: *High Orbit Ion Cannon - HOIC* é usada pelo grupo *hacker Anonymous* e gera grande quantidade de requisições HTTP, possuindo interface gráfica. Utilizou-se contra o departamento de Justiça dos EUA na época da decisão de tirar do ar o portal Megaupload.com (RADWARE, 2016b), (KUMAR et al., 2016b);

f) *Slowloris*: utilizada em um tipo de ataque *LR DDoS* homônimo que opera na camada de aplicação do modelo OSI, que exauri as conexões de um servidor *web* com requisições HTTP GET incompletas. (RADWARE, 2016b), (THOMSON; CONRAD, 2011), (ZLOMISLIĆ et al., 2017);

g) *Saphyra*: empregada em ataques HTTP *Flood* (MCMILLEN, 2016);

h) *Sockstress*: ferramenta para a geração de ataques *sockstress* desenvolvida em C e com versão em Python (HORNBY, 2012), (HORNBY, 2014);

i) *Trinoo* ou *Trin00*: para ataques *UDP flood* com pacotes UDP de tamanho constante contra portas aleatórias da vítima (KUMAR et al., 2016b);

j) *Knight*: baseada em IRC e utilizada em ataques UDP *flood* e SYN *flood* (KUMAR et al., 2016b), (ALOMARI et al., 2012);

k) *Tribe Flood Network* (TFN): pode efetuar ataques *smurf*, UDP *flooding*, TCP *SYN flooding*, ICMP *echo request flooding* (ALOMARI et al., 2012);

l) *RUDY (R-U-Dead-Yet?)*: baseada em IRC e efetua ataques *LR DDoS* com HTTP *post request* (KUMAR et al., 2016b);

m) DDOSIM: ferramenta para a geração de ataques HTTP *post request* (KUMAR et al., 2016b).

TAB. 9.2: Ferramentas de geração de ataques *DDoS*

Ferramenta	Tipo de Ataque	Principais Caracteristicas	Observações
SlowHTTPtest	Ataques *Slow* HTTP (*slowloris*, *slow read* e *slow body*).	*opensource* e escrita em C++.	–
LOIC	HTTP request floodig.	IP não *spoofed*.	IRC.
THC-SSL-DoS	explora a assimetria do protocolo SSL.	inicia um *handshake* SSL e, imediatamente, solicita a renegociação da chave, repetindo inúmeras vezes.	roda em windows e Linux .
HOIC	HTTP request floodig.	interface gráfica.	IRC.
Slowloris	*Slowloris*.	*LR DDoS*.	*script* em Perl.
Saphyra	HTTP *Flood*.	-	é um *script* Python.
Sockstress	*Sockstress*.	*LR DDoS*.	código-fonte em C e Python.
Trinoo	*UDP flood*.	pacotes UDP de tamanho constante contra portas aleatórias da vítima	*telnet*.
Knight	UDP *flood* e SYN *flood*.	escrito em C++ .	IRC.
TFN	*smurf*, UDP *flooding*, TCP *SYN flooding*, ICMP *echo request flooding*.	–	conexão TCP.
RUDY	*LR* HTTP *post request*.	*LR DDoS*.	IRC.
DDOSIM	*H*TTP *post request*.	escrito em C++.	conexão TCP encriptada.

APÊNDICE 3: LABORATÓRIO DE ATAQUES *DDOS*: A DESCRIÇÃO DO EXPERIMENTO

Os servidores *web* mais utilizados são o Apache, o IIS e o nginx (TRIPATHI et al., 2016). Neste trabalho foi utilizada a versão 2.4.10 do Apache.

Para atingir os objetivos desta dissertação, foi criado um ambiente virtualizado com o *software* Vmware Workstation no laboratório de Programação de Computadores da SE/8, do Instituto Militar de Engenharia para a simulação de ataques DDoS a um servidor *web* Apache. A configuração do laboratório é a seguinte:

- 11 equipamentos *Desktops* Dell *All-in-One*;
- Processador Intel core i7-4770S;
- 8GB de memória RAM;
- Disco Rígido de 1 TB;
- Sistema Operacional Ubuntu 14.01 64 *bits*;
- Rede LAN cabeada; e
- *Switch* Catalyst 2960.

Foram empregadas 11 máquinas físicas, sendo 10 atacantes, cada uma com 7 máquinas virtuais Debian Linux versão 8.7 32 bits, e gerando tráfegos *slowloris* e *sockstress*. Em um dos equipamentos, foi instalado o servidor *web* virtual com Debian Linux versão 8.7 32 *bits*.

O tráfego de fundo foi gerado com o wget[1] configurado no crontab[2] do Linux e com a ferramenta httpmon, que gera tráfego HTTP com intervalo entre duas requisições consecutivas seguindo uma distribuição exponencial (GRIMALDI et al., 2015).

A coleta do tráfego de rede em formato PCAP (*sniffer*) foi efetuado com a ferramenta tcpdump, na máquina física contendo o servidor *web*. Os relógios internos dos equipamentos foram sincronizados com o protocolo NTP.

[2]agendador de tarefas

[2]utilitário para o *download* de arquivos em páginas *web*

As placas de rede das máquinas virtuais foram configuradas em modo *bridge* com a faixa de IP 192.168.91.X, máscara de rede 255.255.255.0, sendo 192.168.91.5 o IP do servidor *web*. Cada uma das máquinas virtuais atacantes com o sistema Debian foi configurada com 600 MB de memória RAM e o servidor *web* virtual, 1.2 GB de memória RAM e 100 GB de HD. As informações referentes as configurações das máquinas virtuais estão descritas na tabela na tabela 9.3.

TAB. 9.3: Configurações das máquinas virtuais

	Servidor Web	Atacantes
Memória RAM	1.2 GB	600 MB
HD	100 GB	100 GB
Faixa de IP	192.168.91.5	192.168.1.1/24
Sistema Operacional	Debian 8.7 32 *bits*	Debian 8.7 32 *bits*
Total	1 Servidor Apache	70 atacantes

A ferramenta *opensource* slowhttptest (TRIPATHI et al., 2016), que vem instalada por padrão em muitas distribuições Linux, foi utilizada para gerar os ataques. O slowhhtptest permite testar servidores *web* a fim de buscar vulnerabilidades a ataques *DDoS* ou simplesmente checar quantas conexões simultâneas o servidor pode gerenciar. Pode-se, inclusive, iniciar um ataque utilizando o slowhttptest a partir de um *smartphone*.

Embora os ataques estivessem programados para iniciar em um horário fixo, estes duravam no máximo 240 segundos e, era aguardado um tempo aleatório entre 180 segundos e 250 segundos para começarem novamente.

O tráfego de rede capturado no formato PCAP e coletado em três dias consecutivos está distribuído conforme a tabela 9.4 e foi constituído de tráfego de ataques (ataques *slowloris* e *sockstress*) e tráfego de fundo (simulação de acesso produzido por usuários legítimos a *sites web* e tráfego gerado pela ferramenta httpmon).

TAB. 9.4: Distribuição do tráfego capturado

Horário	Conteúdo
07:50:00	Apenas Tráfego de Fundo.
08:50:00	Início do Ataque *sockstress*.
09:30:00	Fim do Ataque *sockstress*.
09:40:00	Início do Ataque *slowloris*.
10:20:00	Fim do Ataque *slowloris*.

O *dataset* contém, além dos *dumps* em formato PCAP, os seguintes arquivos:

- *logs* do servidor Apache (access.log e error.log);
- as informações do estado da memória (total, livre, usada, *buffers* e cache) e CPU durante todo o experimento (formatos txt , XML) ;
- os metadados dos ataques em formato XML.

Os valores de utilização de CPU, memória e rede (dados recebidos e transmitidos) do servidor *web* foram monitorados antes e durante os ataques com a ferramenta Sysstat do Linux, que permite a geração de relatórios nos formatos XML e JSON. Em relação a CPU, são armazenados o tempo do usuário (us), o tempo de CPU dedicado aos códigos que não são do *kernel*, tempo de sistema (sy), que é o tempo dedicado aos códigos do *kernel*, tempo ocioso do sistema (id) e o tempo de espera por operações de entrada e saída (wa). As estatísticas utilizadas para avaliar o impacto dos ataques no servidor *web* são a média do percentual de CPU ociosa (CPU i), o seu desvio padrão (CPU sd), a média de memória utilizada em KB (Mem), o seu desvio padrão (Mem sd), a média do total de KB recebidos por segundo (rxkB), o seu desvio padrão (rxkB sd), a média do total de KB transmitidos por segundo (txkB) e o seu desvio padrão (txkB sd). Os resultados são apresentados nas tabelas 9.5 e 9.6 e nos gráficos 9.1 e 9.2, 9.3, 9.4, 9.5 e 9.6. As análises estão na seção 9.4.

TAB. 9.5: Estatísticas 1

	CPU i	CPU sd	Mem	Mem sd
Sem Ataques	98%	0,90	285642	73700
Slowloris	63%	3,08	363621	25716

TAB. 9.6: Estatísticas 2

	rxkB	rxkB sd	txkB	txkB sd
Sem Ataques	10	2,03	8,54	1,88
Slowloris	108	21,53	70,73	13,77

O servidor *web* utilizado foi Apache, com as configurações de *hardening*, instalando em VMware Workstation executando Debian. As máquinas atacantes também utilizaram o sistema operacional Debian .

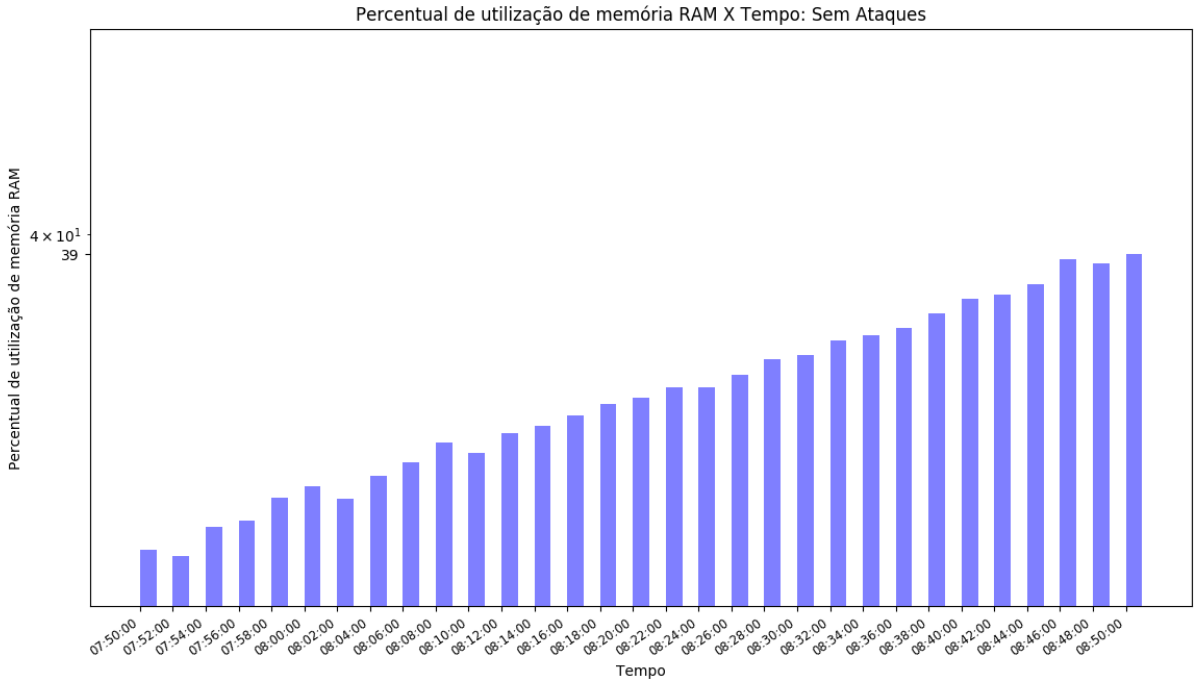

FIG. 9.1: Percentual de utilização de memória RAM X Tempo: Sem Ataques

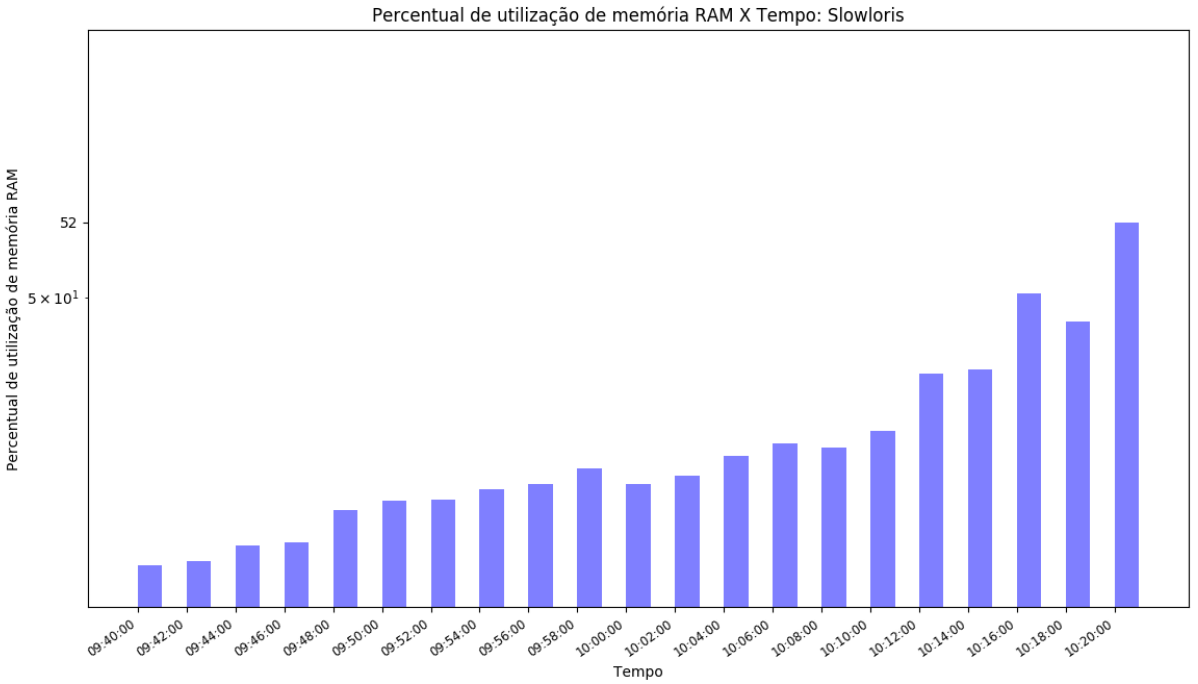

FIG. 9.2: Percentual de utilização de memória RAM X Tempo: *Slowloris*

O tráfego de fundo foi gerado por uma máquina virtual Debian com wget acessando uma lista de 60 páginas *web* agendadas no crontab e pela ferramenta httpmon instalada em cada um dos atacantes. O tráfego de rede coletado com tcpdump (arquivo PCAP) foi submetido a análises afim de validar ou não a hipótese desta dissertação.

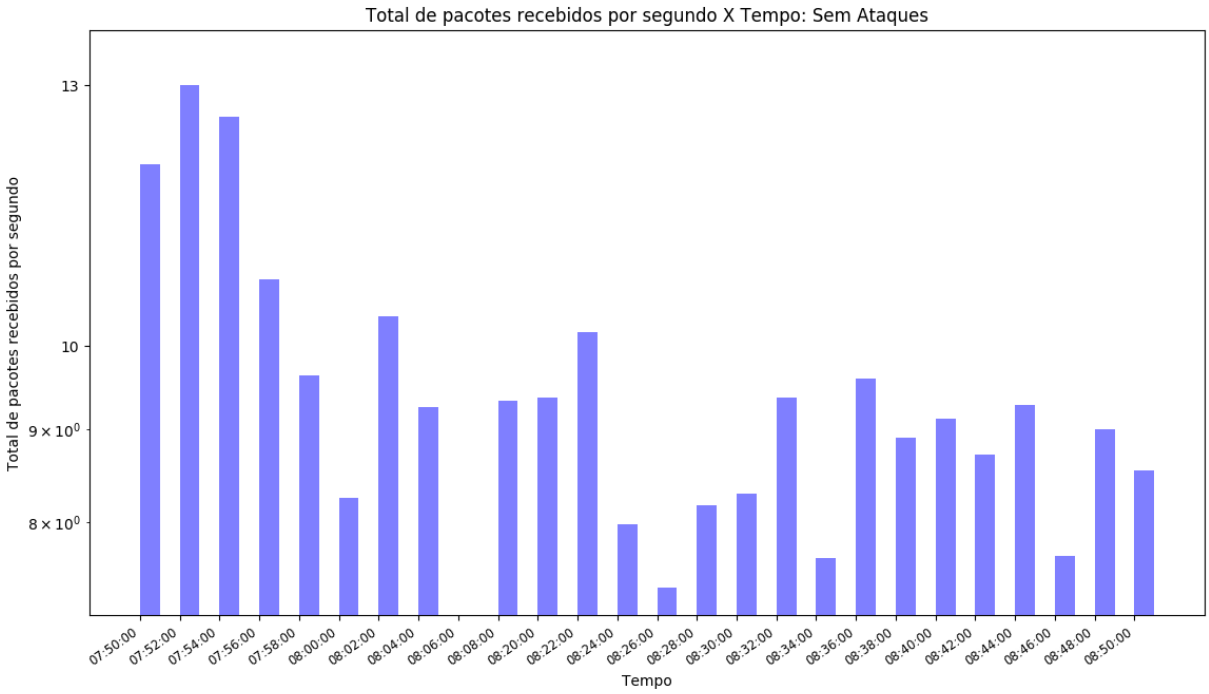

FIG. 9.3: Total de pacotes recebidos por segundo X Tempo: Sem Ataques

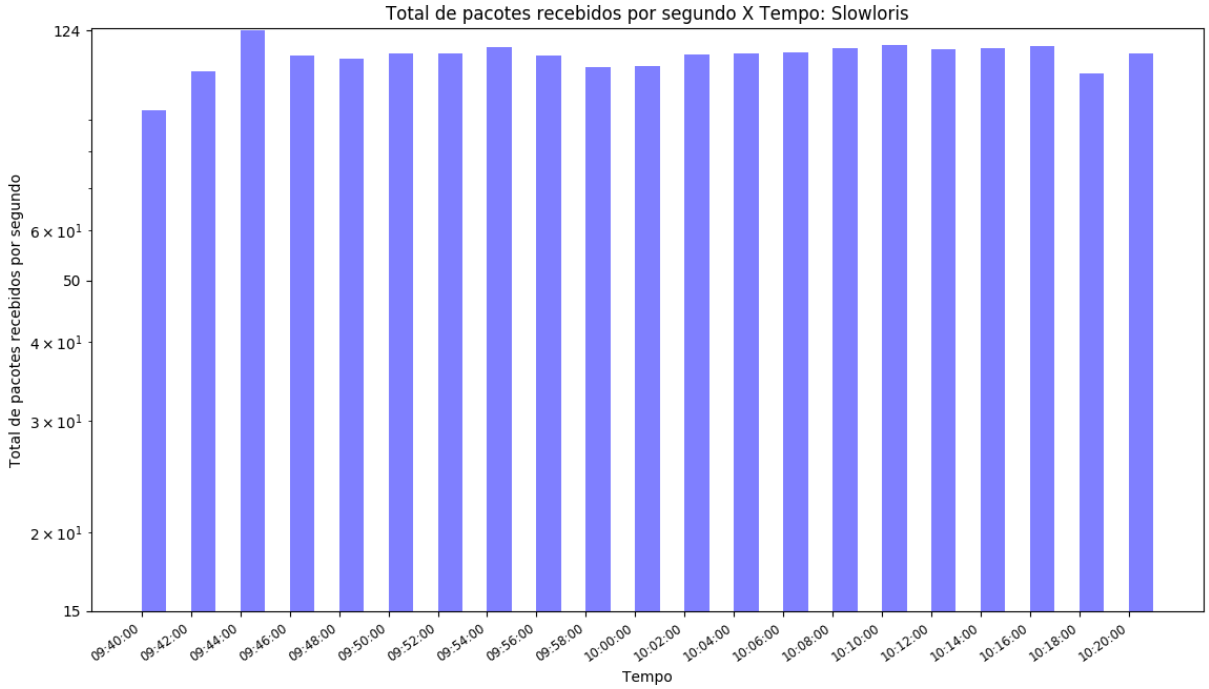

FIG. 9.4: Total de pacotes recebidos por segundo X Tempo: *Slowloris*

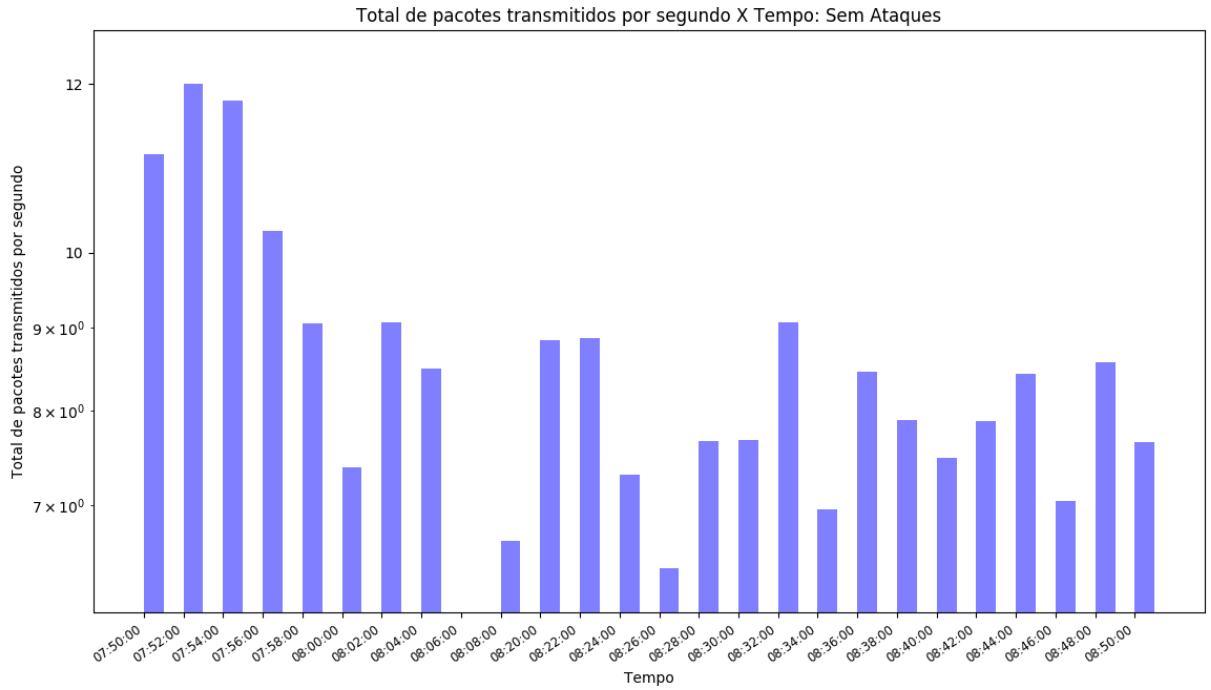

FIG. 9.5: Total de pacotes transmitidos por segundo X Tempo: Sem Ataques

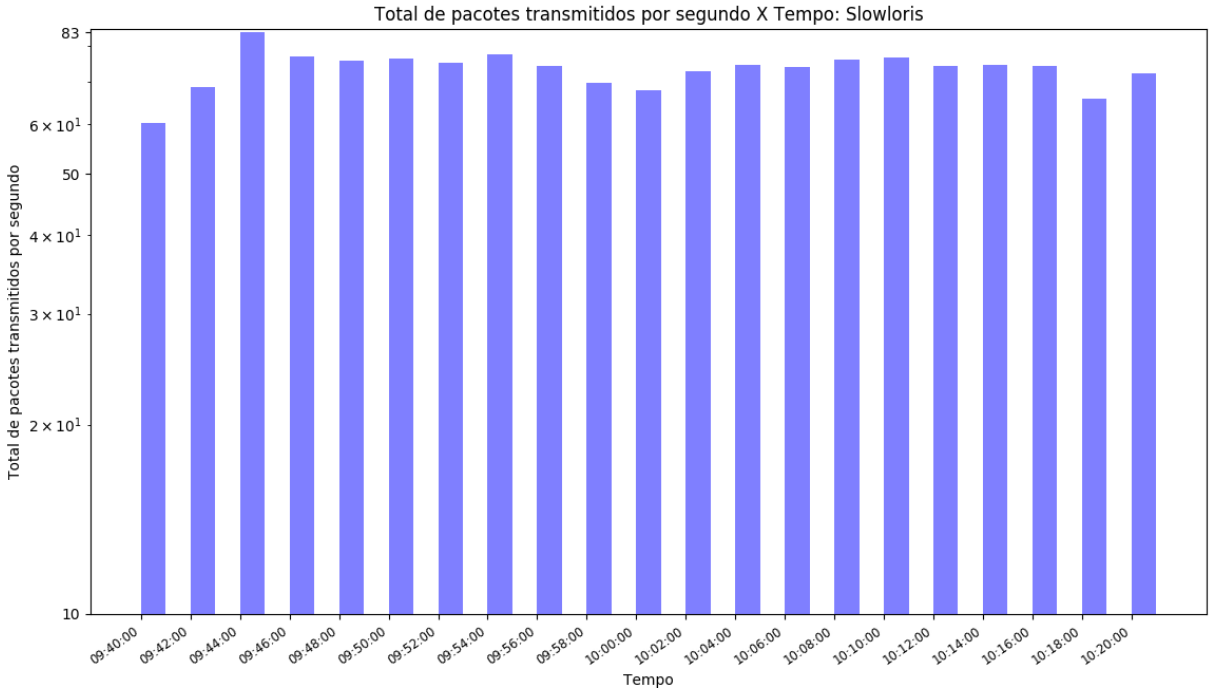

FIG. 9.6: Total de pacotes transmitidos por segundo X Tempo: *Slowloris*

Foi efetuada a comparação entre o comportamento de pacotes de rede antes e depois da ocorrência de um ataque *LR DDoS* para confirmar a eficência dos ataques de negação de serviço.

Antes do início do ataque *slowloris*, o arquivo access.log contido no diretório /var/-log/apache2 exibia o código 200 do protocolo HTTP, o que apontava o sucesso na requisição ao servidor. Após o início do ataque, o código 408 foi a resposta do servidor, ou seja, ocorreu o estouro do temporizador (*timeout*) das requisições. Após o fim do ataque, o código de resposta do protocolo HTTP voltou a ser o 200.

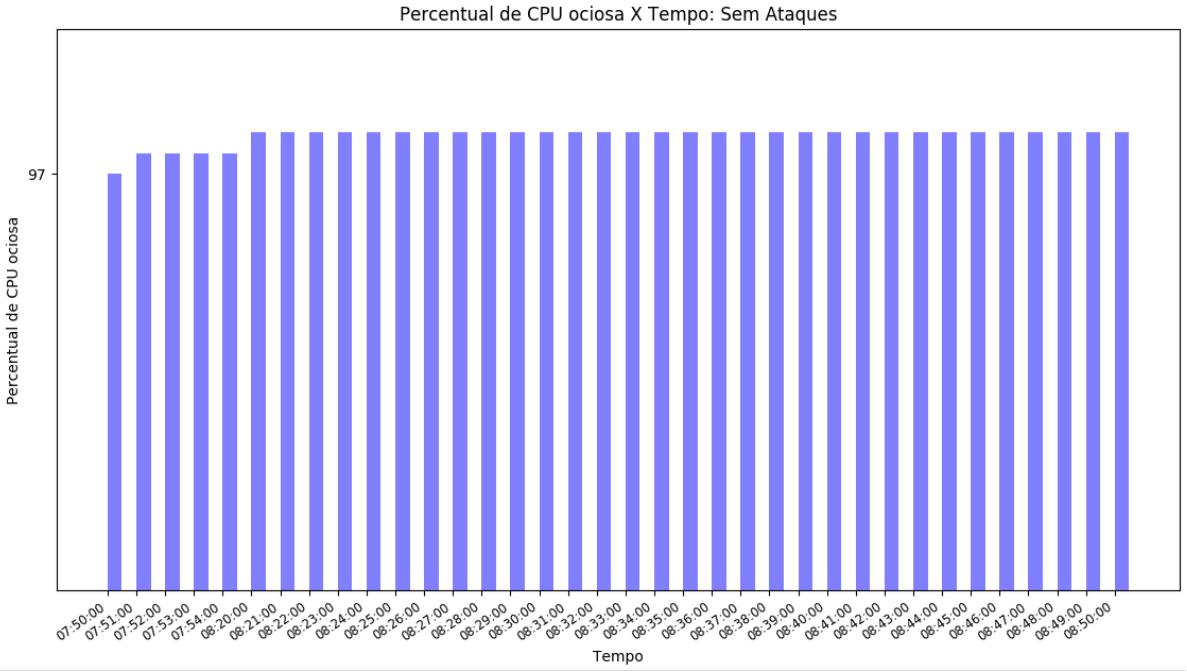

FIG. 9.7: Percentual de CPU ociosa X Tempo: Sem Ataques

Observando os números apresentados nas tabelas 9.5 e 9.6, percebe-se que o ataque *slowloris* foi efetivo em aumentar a utilização dos recursos do servidor *web*. A máquina virtual que continha o servidor *web* Apache teve um aumento de 35% na média de utilização de CPU (figuras 9.7 e 9.8), 10% de aumento na média de utilização de memória (figuras 9.1 e 9.2), a média de pacotes recebidos aumentou de 10,8 vezes e a média de pacotes enviados aumentou de 8,3 vezes durante o ataque *slowloris*. Pode-se verificar o aumento do tempo de resposta do servidor, conforme as figuras 9.9 e 9.10.

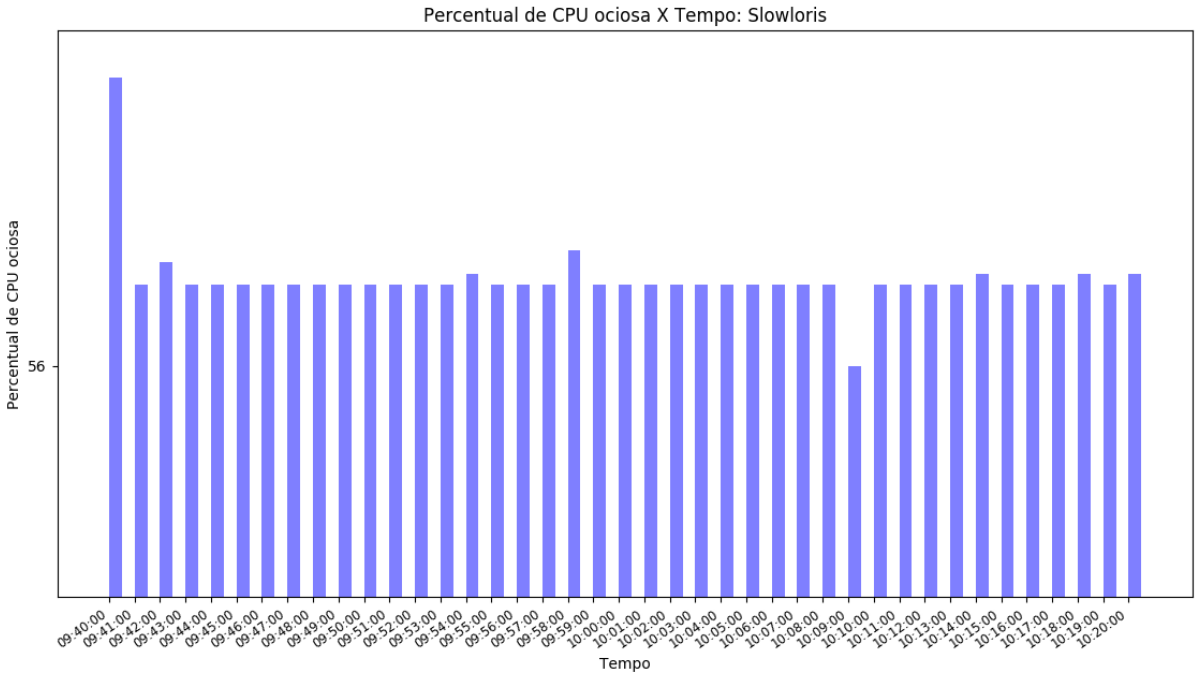

FIG. 9.8: Percentual de CPU ociosa X Tempo: *Slowloris*

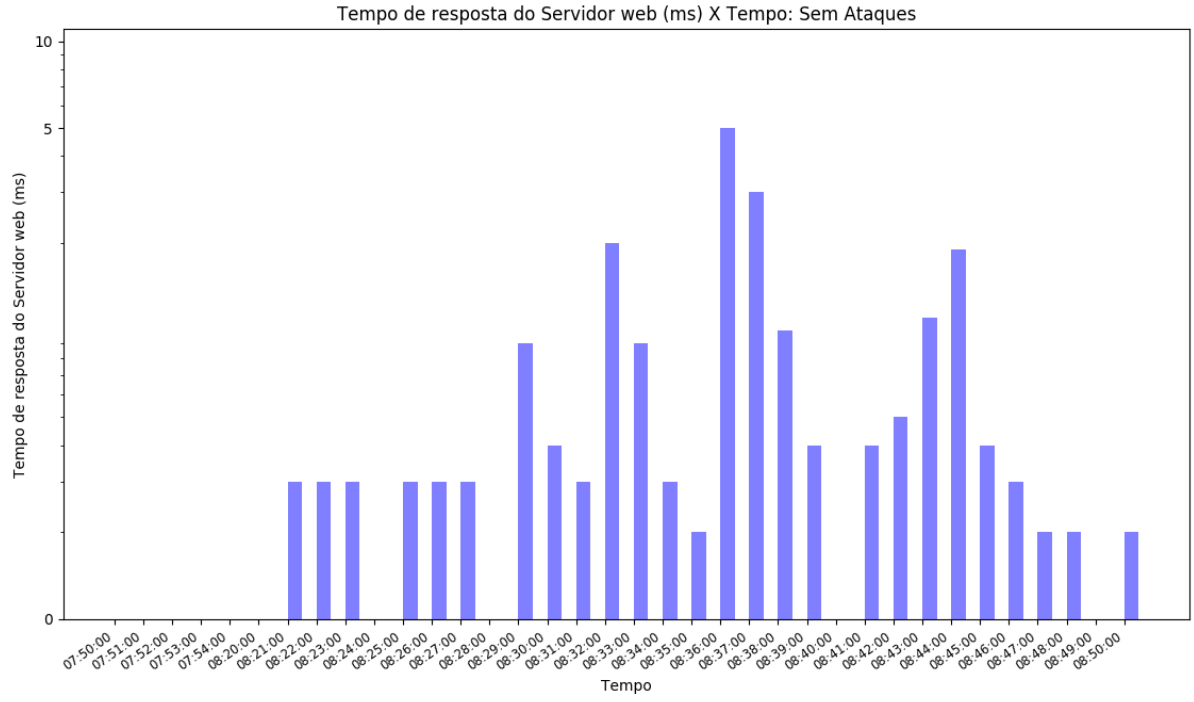

FIG. 9.9: Tempo de Resposta do Servidor web (em ms) X Tempo: Sem Ataques

Ao observar o *dump* do ataque *slowloris*, nota-se a regularidade dos tamanhos das janelas do TCP (os valores são sempre os mesmos), conforme a figura 9.14, comparando-se com os tamanhos das janelas no tráfego normal (figura 9.13). A figura 9.15 mostra o reduzido tamanho de janela durante o ataque *sockstress*.

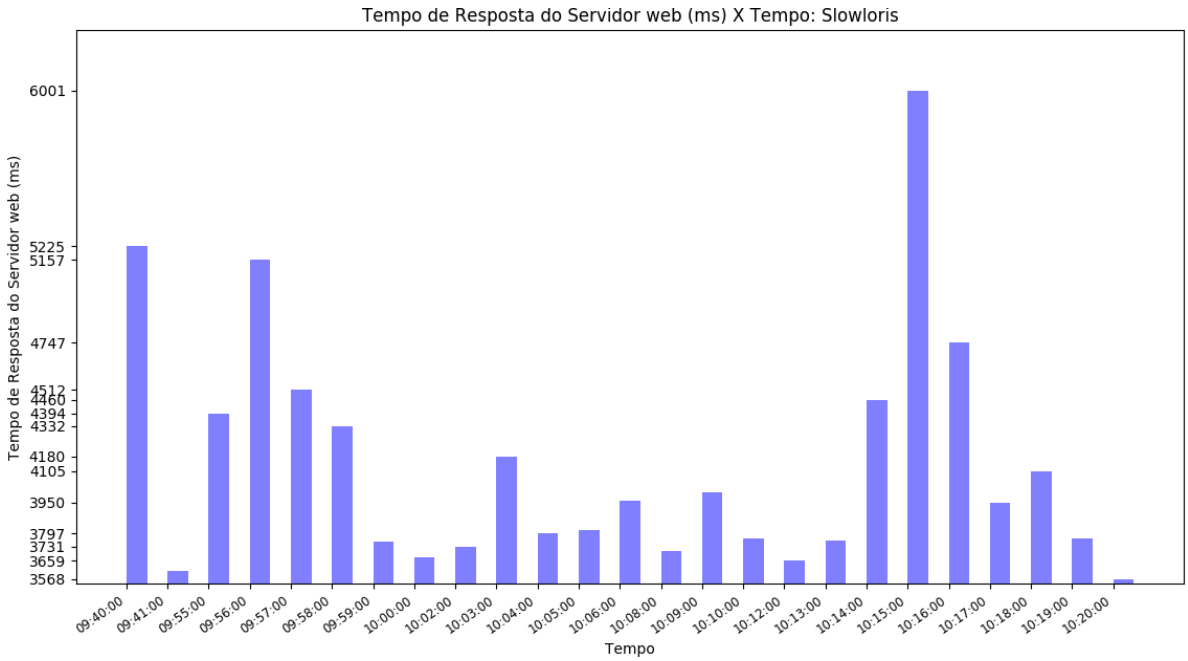

FIG. 9.10: Tempo de Resposta do Servidor web (em ms) X Tempo: *Slowloris*

Antes do início dos ataques, a média do número de conexões por segundo era de 19 conexões por segundo, conforme a figura 9.11. Contando-se o número de conexões diferentes abertas por IP 192.168.91.5 (servidor *web*) na porta 80 por segundo, durante o ataque *slowloris*, em cada *time slice* de 1 minuto, tem-se uma média de 72 conexões por segundo, conforme a figura 9.12. Análise similar pode ser efetuada com o ataque *sockstress*, resultando em 52 conexões por segundo. Uma instalação padrão do Apache permite no máximo 150 conexões simultâneas (variável MaxRequestWorkers), conforme a documentação oficial.

Os experimentos demonstraram que o ataque *slowloris* é efetivo em consumir os recursos de um servidor *web* Apache, especialmente a memória, e torná-lo indisponível. Com 70 máquinas virtuais atacantes, foram geradas em média 72 conexões por segundo durante o ataque *slowloris* e o servidor Apache teve um incremento de 8% na utilização de CPU, 27,3% em memória, 10,8 vezes em pacotes recebidos e 8,3 vezes em pacotes enviados. Como os ataques do tipo *slow* HTTP geram um tráfego quase idêntico a um legítimo, sua detecção torna-se difícil para *firewalls*.

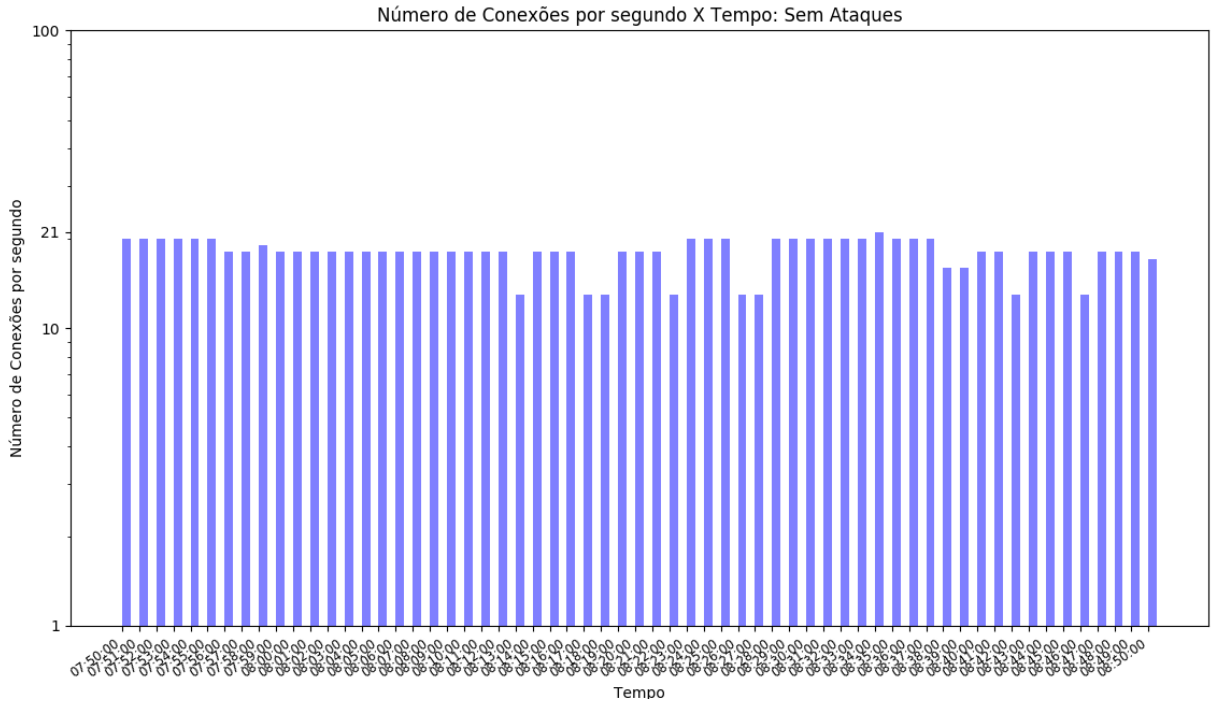

FIG. 9.11: Número de Conexões/segundo Sem Ataques

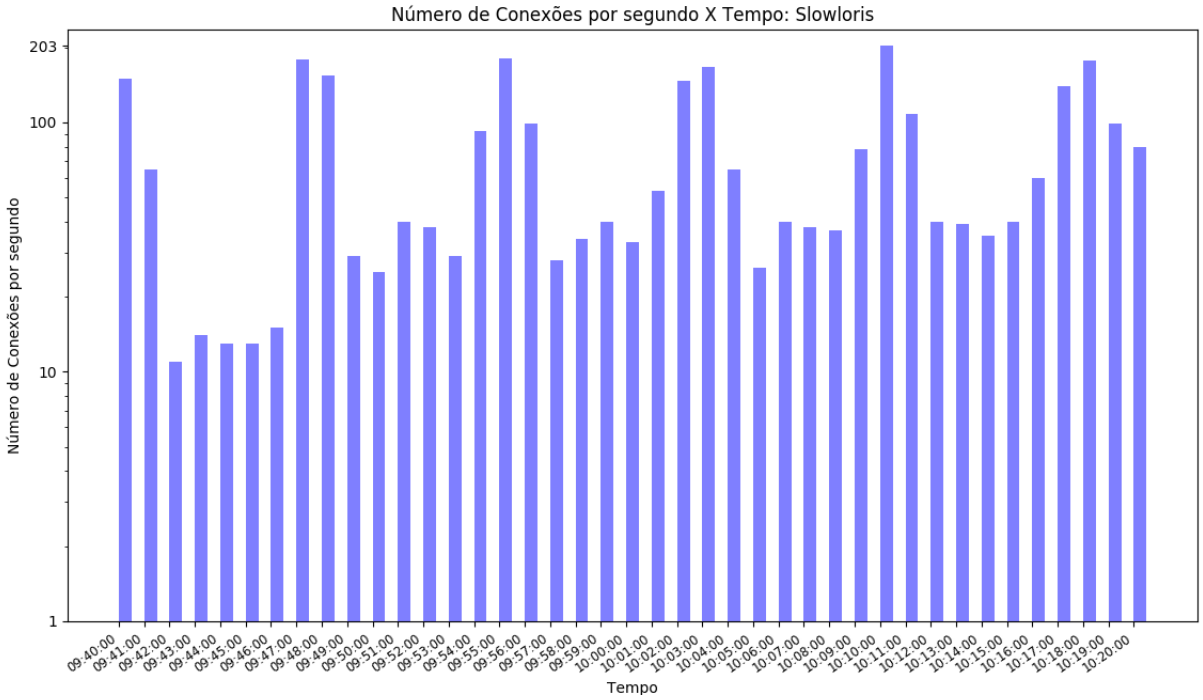

FIG. 9.12: Número de Conexões/segundo com ataque *Slowloris*

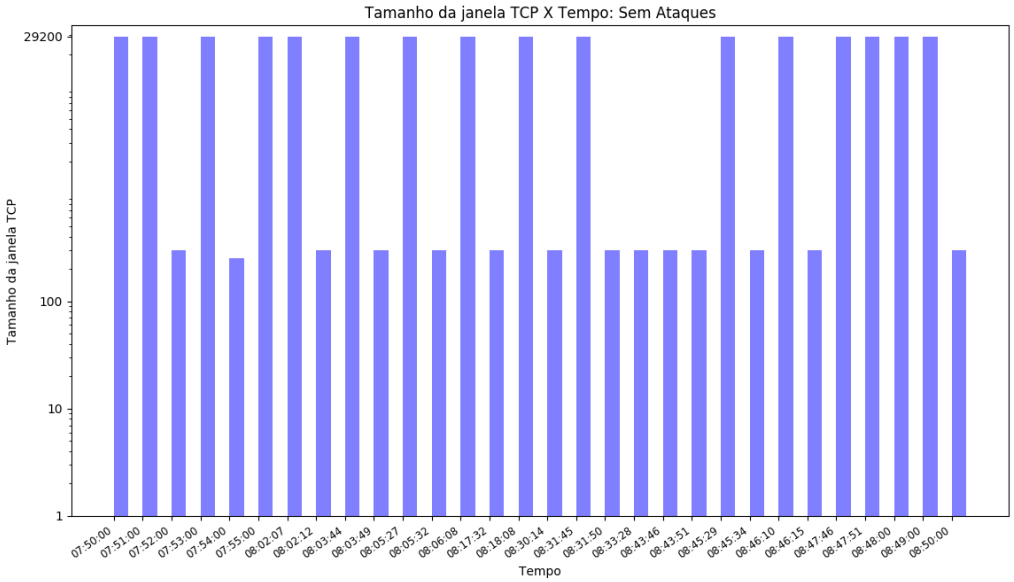

FIG. 9.13: Tamanho da janela X tempo (sem Ataques)

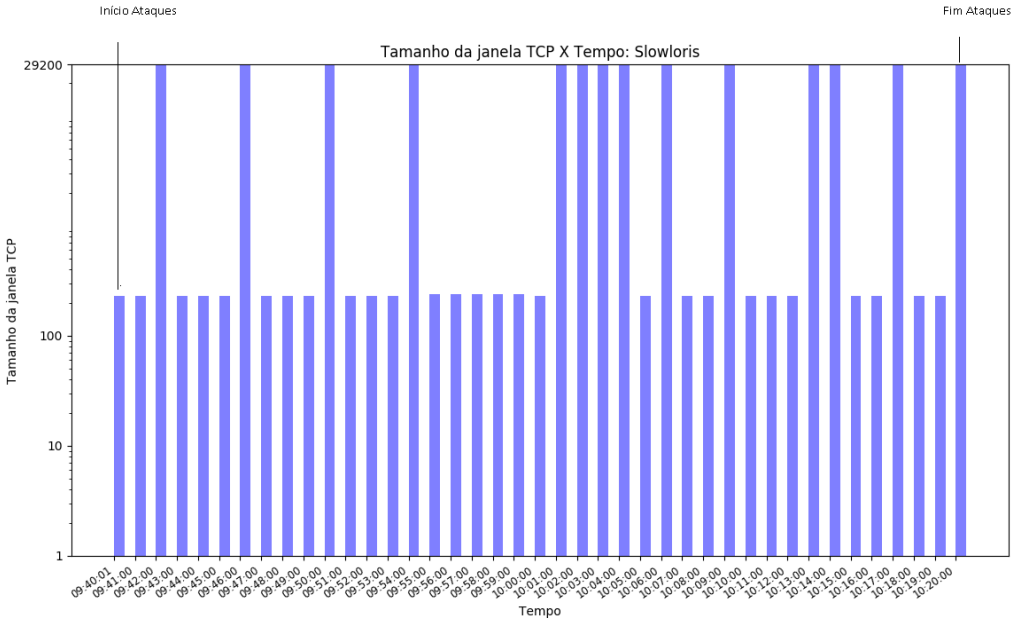

FIG. 9.14: Tamanho da janela X tempo (*slowloris*)

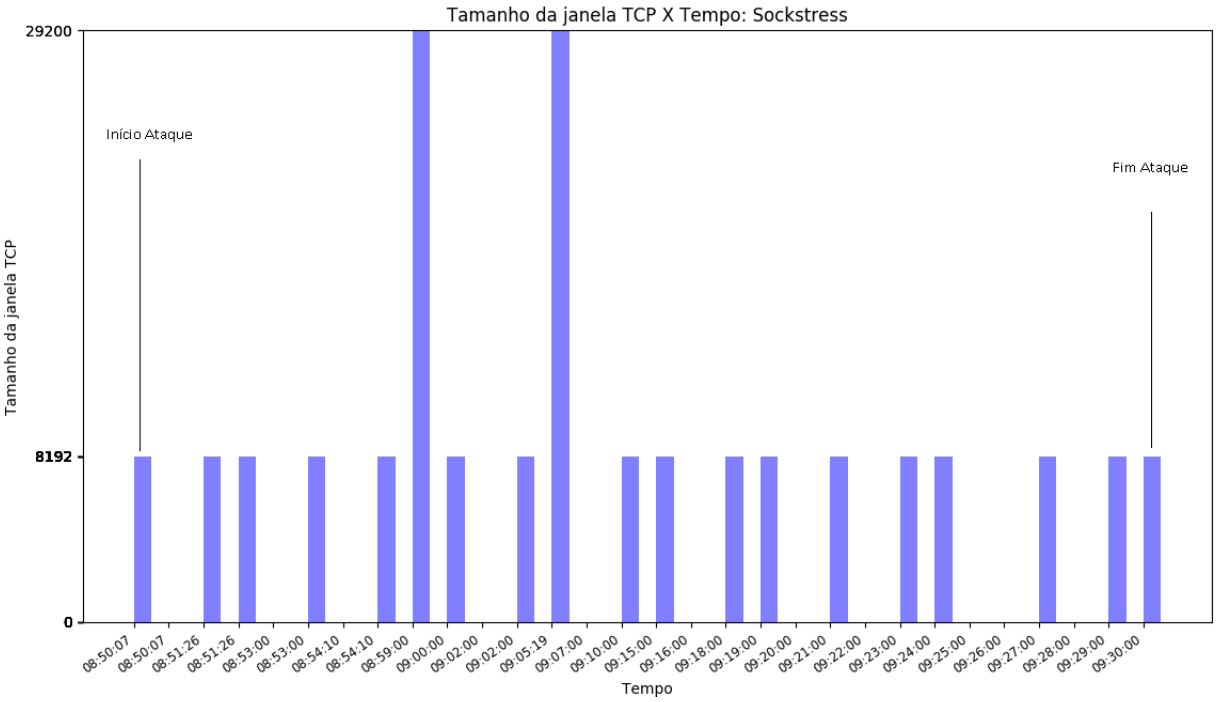

FIG. 9.15: Tamanho da janela X tempo (*sockstress*)

APÊNDICE 5: O CÁLCULO DA ENTROPIA: UM EXEMPLO

Para o presente exemplo, considere os endereços IPs de dois clientes 192.168.1.4 e 192.168.1.8 e um servidor *web* cujo IP é 192.168.1.6. A função $Separa\_IP\_Origem\_Destino$ do algoritmo Detecta *Slowloris*, lerá o *slice* de tempo, buscará pelos *handshakes* TCP entre o servidor *web*, conforme a figura 9.16, e os diversos clientes, gerando dois arquivos: um com IPs de origem e outro IPs de destino ($ArquivoIPs$), resultando na tabela 9.7.

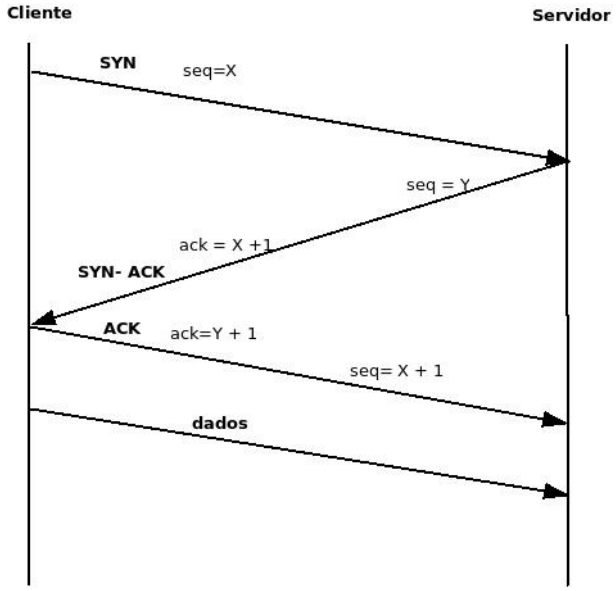

FIG. 9.16: *Handshake TCP*

Aplicando-se a função $Calcula\_entropia\_IP\_Or\_Dst$ nos arquivos $ArquivoIP[Or]$ e $ArquivoIP[Dst]$ anteriores, calcular-se-a a entropia de IP de origem (ou IP destino) segundo a equação 9.1, com $n = 6$.

TAB. 9.7: ArquivoIP

ArquivoIP[Or]	ArquivoIP[Dst]
192.168.1.4	192.168.1.6
192.168.1.6	192.168.1.4
192.168.1.4	192.168.1.6
192.168.1.8	192.168.1.6
192.168.1.6	192.168.1.8
192.168.1.8	192.168.1.6

TAB. 9.8: Frequências dos IPs

IP[Or]	Freq	IP[Dst]	Freq
192.168.1.4:	1/3	192.168.1.4:	1/6
192.168.1.6:	1/3	192.168.1.6:	2/3
192.168.1.8:	1/3	192.168.1.8:	1/6

$$S = -\sum_{i=1}^{n} W \log_2(W) \tag{9.1}$$

As frequências de ocorrência de cada um dos endereços IPs está na tabela 9.8. O valor da entropia de *Shannon* dos IPs de origem e IPs de destino será, em *bits*, respectivamente:

$$-(\frac{1}{3} * \log_2 \frac{1}{3} + \frac{1}{3} * \log_2 \frac{1}{3} + \frac{1}{3} * \log_2 \frac{1}{3}) = 1.58$$

$$-(\frac{1}{6} * \log_2 \frac{1}{6} + \frac{2}{3} * \log_2 \frac{2}{3} + \frac{1}{6} * \log_2 \frac{1}{6}) = 1.25$$

APÊNDICE 6: COMANDOS DO LABORATÓRIO DE ATAQUES DDOS

Captura com tcpdump:

tcpdump -i wlan0 -v -w arquivo.cap

Disparar ataque sockstress:

./sockstress IP:80 eth0

Disparar ataque slowloris:

python3 slowloris.py IP webServer

Disparar ataque TCP SYN flood com hping3:

hping3 -i u1 -S -p 80 IP webServer

APÊNDICE 7: SEPARAÇÃO DE IP DE DESTINO E IP DE ORIGEM EM PYTHON

```
from scapy.all import *


pkts = rdpcap("slice.pcap")

for p in pkts:

        if IP in p: #if packet has IP layer
                src_ip = p[IP].src
                dest_ip = p[IP].dst
                #print src_ip

                f = open('IP_src.txt', 'a')
                for ip in src_ip:
                        f.writelines(ip)
                        f.write("\n")
                        f.close()

                f = open('IP_dest.txt', 'a')
                for ip in dest_ip:
                        f.writelines(ip)
                        f.write("\n")
                        f.close()
```

APÊNDICE 8: CÁLCULO DA ENTROPIA DE SHANNON EM PYTHON

```
# -*- coding: utf-8 -*-

import numpy as np
import collections
import os

f = open("IP_dest.txt", 'r')
sample_ips = f.readlines()
C = collections.Counter(sample_ips)
counts = np.array(list(C.values()),dtype=float)
#counts  = np.array(C.values(),dtype=float)
prob    = counts/counts.sum()
shannon_entropy = (-prob*np.log2(prob)).sum()
print ("A entropia dos IP de destino e",shannon_entropy)
fdest = open('Entropia_IP_dst.txt','a') ###modo append
fdest.write(str(shannon_entropy))
fdest.write('\n')
fdest.close()
f.close()
os.remove("IP_dest.txt")##apagar os arquivos IP_src.txt e IP_dest.txt
#para calcular nova entropia

f = open("IP_src.txt", 'r')
sample_ips = f.readlines()
C = collections.Counter(sample_ips)
counts = np.array(list(C.values()),dtype=float)
prob    = counts/counts.sum()
shannon_entropy = (-prob*np.log2(prob)).sum()
print ("A entropia dos IP de origem e",shannon_entropy)
fdest = open('Entropia_IP_src.txt','a')
```

```
fdest.write(str(shannon_entropy))
fdest.write('\n')
fdest.close()
os.remove("IP_src.txt")#apagar os arquivos IP_src.txt e IP_dest.txt
#para calcular nova entropia
```

APÊNDICE 9: CÁLCULO DO PVS E FCS

```
#include <pcap.h>
#include <net/ethernet.h>
#include <netinet/ip.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <arpa/inet.h>
#include <string.h>

int iCount = 0;
int iCount1 = 0;
struct timeval  start_ts, end_ts;

void packetHandler(u_char *userData, const struct pcap_pkthdr* pkthdr, cons

int main(int argc, char **argv) {
pcap_t *descr;
char errbuf[PCAP_ERRBUF_SIZE];

//abrir PCAP para processamento offline
descr = pcap_open_offline(argv[1], errbuf);
if (descr == NULL) {
printf("pcap_open_live() failed:\n");
return 1;
}


if (pcap_loop(descr, 0, packetHandler, NULL) < 0) {
printf("pcap_loop() failed: ");
return 1;
}
```

```
printf("Pacotes com \\r\\n\\r\\n : %d\n", iCount);
printf("Tempo total por PCAP : %ld\n", end_ts.tv_sec-start_ts.tv_sec);
printf("Media de pacotes por segundo : %f\n",(float) (iCount/(float)((end_t
printf("Pacotes com payload zero :  %d\n", iCount1);
printf("Pacotes com payload zero por segundo:  %f\n",(float) (iCount1/(floa
return 0;
}

void packetHandler(u_char *userData, const struct pcap_pkthdr* pkthdr, cons
const struct ether_header* ethernetHeader;
const struct ip* ipHeader;
const struct tcphdr* tcpHeader;
char sourceIp[INET_ADDRSTRLEN];
char destIp[INET_ADDRSTRLEN];
u_int sourcePort, destPort;
u_char *data;
int dataLength = 0;
//string dataStr;
int i =0;

if( iCount == 0) {
start_ts = pkthdr->ts;
}

ethernetHeader = (struct ether_header*)packet;
if (ntohs(ethernetHeader->ether_type) == ETHERTYPE_IP) {
ipHeader = (struct ip*)(packet + sizeof(struct ether_header));
inet_ntop(AF_INET, &(ipHeader->ip_src), sourceIp, INET_ADDRSTRLEN);
inet_ntop(AF_INET, &(ipHeader->ip_dst), destIp, INET_ADDRSTRLEN);

if (ipHeader->ip_p == IPPROTO_TCP) {
tcpHeader = (struct tcphdr*)(packet + sizeof(struct ether_header) + sizeof(
sourcePort = ntohs(tcpHeader->source);
```

```
destPort = ntohs(tcpHeader->dest);
data = (u_char*)(packet + sizeof(struct ether_header) + sizeof(struct ip) +
dataLength = pkthdr->len - (sizeof(struct ether_header) + sizeof(struct ip)




//printf("%d.%d\n", pkthdr->ts.tv_sec, pkthdr->ts.tv_usec);
#if 1
if(destPort == 80) {
for (i = 0; i < dataLength; i++) {
//                          if(data[i] == 13 && data[i+1] == 11 && data[i+2] ==
if((data[i] == '\r') && (data[i+1] == '\n') && data[i+2] == '\r' && data[i+
{
iCount++;
break;
}
}
}
#endif
if((destPort == 80) && (dataLength == 0)) {
iCount1 ++;
}
}
end_ts = pkthdr->ts;
}
}
```