Ferramenta para Detecção e Contenção de Ataques Slowloris

Vinicius S. Faria[1], **Jéssica A. Gonçalves**[1] **Camilla A. M. Silva**[1] **Gabriele B. Vieira, Dalbert M. Mascarenhas**[1]

[1]Centro Federal de Educação Técnlógica Celso Suckow da Fonseca (CEFET)
Rua do Imperador - Centro - Petrópolis - RJ - Brasil

{vfaria,jalcantara,gbritto,calves}@e-computacao.com.br

dalbert.mascarenhas@cefet-rj.br

Abstract. *Denial-of-service attacks at the application layer bring an additional challenge in their detection by security tools due to the use of application-specific vulnerabilities. This work presents a tool for detection and contention of Slowloris attack. The present tool operates in three distinct modules, distributed in routers, WEB servers, and in a Controller host. The modules act together to detect and contain the attack by analyzing the behavior pattern of the attack. Laboratory tests demonstrate that the tool detects the anomalous behavior of Slowloris and consequently blocks the attacker before the saturation of the victim's resources.*

Resumo. *Ataques de negação de serviço na camada de aplicação trazem um desafio adicional em sua detecção por ferramentas de segurança devido a utilização de vulnerabilidades específicas da aplicação. Este trabalho apresenta uma ferramenta para detecção e contenção de ataque Slowloris. A ferramenta atua por meio de três módulos distintos, distribuídos nos roteadores, servidores WEB e em um Concentrador. Os módulos atuam em conjunto para detectar e conter o ataque através da análise do padrão de comportamento do ataque. Os testes realizados em laboratório demonstram que a ferramenta detecta o comportamento anômalo do Slowloris e consequentemente bloqueia o atacante antes da saturação dos recursos da vítima.*

1. Introdução

Atualmente ataques de negação de serviço DoS (Denial of Service) têm representado uma grande ameaça para os recursos disponibilizados na Internet e nas redes internas. Este tipo de ataque tem como objetivo impossibilitar o acesso de usuários legítimos a recursos disponibilizados na rede [da Silva et al. 2016]. O ataque geralmente é executado através da exploração de vulnerabilidades do alvo ou da exaustão dos recursos do mesmo, que podem ser relacionados a redes ou a performance de serviços [Gu and Liu 2007, Corrêa and Martins 2013]. Parte dos ataques de DoS são direcionados a servidores de aplicação que mantêm serviços WEB e e-mail, o objetivo é tornar estes serviços inacessíveis na rede após o esgotamento dos recursos dos hospedeiros das aplicações. Grande parte dos ataques DoS fundamentam-se em estabelecer um grande número de conexões TCP abertas ou semi-abertas no hospedeiro alvo [Goncalves et al. 2017, Yuan et al. 2018]. Por consequência, a máquina provedora do serviço é impossibilitada de aceitar requisições legítimas devido ao grande número de

conexões em espera. Ademais, recursos como memória e processamento também são consumidos durante um ataque. Desta forma, o desafio criado por ataques de DoS em relação ao consumo de serviços e conteúdos tem fomentado estudos que promovem a detecção e a redução dos danos causados por esse tipo de ataque.

Segundo Carl *et al.*, os ataques de DoS direcionados a rede podem ser divididos em ataques de vulnerabilidade e ataques de inundação [Carl et al. 2006]. Nos ataques de inundação, um volume contínuo e excessivo de pacotes é enviado ao alvo, de modo que o tráfego oriundo de fontes seguras é prejudicado com congestionamento e até mesmo descartado. Nos ataques de vulnerabilidade, pacotes mal formados podem ser enviados para as vítimas e estes interagem com alguma falha ou vulnerabilidade existente em alguma aplicação ou recurso hospedado pelo alvo.

Parte dos ataques de DoS podem ser segmentados em ataques que exploram protocolos da camada de Aplicação e são classificados como ADoS (*Application Layer DoS*) [Pascoal et al. 2017]. Dois tipos de ataques ADoS podem ser destacados: *Flooding* e *LowRate*. A utilização do primeiro tipo de ataque produz um excessivo fluxo de tráfego com o objetivo de consumir os recursos da aplicação, até que a mesma fique incapaz de atender novas requisições. O segundo tipo consiste na geração de tráfego semelhante ao de requisições legítimas, utilizando vulnerabilidades encontradas nos protocolos HTTP e HTTPS, de modo a manter requisições em espera por tempo indeterminado [Toklu and Şimşek 2018]. Os ataques do tipo *LowRate*, como o Slowloris, consomem recursos apenas de serviços alvos sem afetar outros recursos disponíveis no hospedeiro.

O presente trabalho apresenta uma ferramenta de detecção e contenção para ataques de vulnerabilidades para servidores de aplicação. Esta ferramenta atua na proteção de ataques oriundos da rede interna, onde o mecanismo de ataque utilizado é o Slowloris [Shorey et al. 2018]. O ataque de Slowloris explora o protocolo HTTP e é capaz de tornar um servidor WEB indisponível com baixa sobrecarga de tráfego [Dantas et al. 2014]. Este tipo de ataque apresenta uma característica semelhante a de um tráfego legítimo. O funcionamento consiste em enviar requisições HTTP GET HEADER incompletas no final do pacote, sempre em um intervalo de tempo, de modo a forçar suas renovações. Dessa forma, o servidor não finaliza as requisições, mantendo-as no *pool* de atendimento até o valor de *timeout* pré-configurado no servidor [Tripathi and Hubballi 2018]. Os recursos são consumidos pelas requisições maliciosas, a medida em que o ataque prossegue realizando as solicitações e mantendo-as abertas. Este comportamento dificulta a detecção do ataque, principalmente porque o consumo se assemelha ao de usuários legítimos. Além disso, durante um ataque bem-sucedido o servidor WEB fica indisponível enquanto a utilização de memória e CPU não apresentam variações significativas, dificultando a detecção do ataque [Singh et al. 2018].

Parte das soluções providas na literatura para o ataque Slowloris utilizam estratégias como limitar o número de conexões para cada usuário ou determinar *timeouts* para cada conexão [Corrêa and Martins 2013, Papadie and Apostol 2017, Barrios Quintanilla and Silva Filho 2017]. Estas limitações podem impedir que usuários legítimos mantenham conexões que superam um limite preestabelecido. Este limite pode ser ultrapassado quando usuários acessam páginas com um alto número de objetos em modo de conexão não persistente do HTTP. Além disso, a limitação do *timeout* provoca a desconexão de usuários que estão com conexões ociosas.

Este trabalho apresenta como contribuição a análise do comportamento do ataque Slowloris e a implementação de uma ferramenta de detecção e prevenção deste tipo de ataque. A solução proposta neste trabalho bloqueia ataques Slowloris sem limitar o número de conexões por clientes e consequentemente prejudicar o acesso por parte de usuários legítimos. A análise é realizada utilizando *logs* do tráfego de rede coletados pelo Tshark [Merino 2013] e analisados com o auxílio da ferramenta Wireshark [Orebaugh et al. 2006]. O objetivo desta análise é detectar comportamentos característicos do ataque abordado. A partir da identificação destas características, são criados filtros que são utilizados para a detecção de um ataque Slowloris. Estes filtros são utilizados nos *logs* das conexões direcionadas ao servidor WEB. Após a correta classificação de um usuário como atacante, a ferramenta proposta bloqueia o endereço MAC do usuário malicioso no roteador sem fio em que este encontra-se diretamente conectado.

A organização do trabalho é descrita a seguir. A seção 2 apresenta os trabalhos relacionados referentes a estratégias de defesa contra ataques Slowloris. A seção 3 descreve a análise do comportamento do ataque e a ferramenta proposta. Os resultados dos experimentos são descritos na seção 4. Posteriormente, a conclusão é apresentada na seção 5.

2. Trabalhos relacionados

Correa *et al.* propõem uma solução para o ataque Slowloris usando o software (D)DoS Deflate [Corrêa and Martins 2013]. O (D)DoS Deflate é um script desenvolvido em Shell Bash utilizado para verificar as conexões que ultrapassam o limite definido para os usuários e consequentemente bloquear o IP referente a essas conexões. O bloqueio pode ser feito utilizando regras de IPTABLES, ou pelo software Advanced Policy Firewall (APF) [Sangeetha]. Esta limitação poderia levar a restringir o acesso de usuários legítimos a páginas com múltiplos objetos em conexão persistente ou restringir clientes que estivessem usando o servidor através de NAT.

Barrios *et al.* propõem um módulo *mod_reqtimeout*, disponibilizado pela Apache Software Foundation, para a mitigação do ataque Slowloris [Barrios Quintanilla and Silva Filho 2017]. O módulo determina o *timeout* e o mínimo de taxa de dados por requisição recebida no servidor. O módulo foi implementado para controlar conexões lentas, buscando mitigar os efeitos de um ataque lento de negação de serviço, como ocorre no Slowloris. A redução do *timeout* proposta pelo módulo *req_timeout* pode gerar falsos positivos caso não haja resposta no tempo de *timeout* estabelecido.

Sousa *et al.* analisam duas ferramentas de IDS (Intrusion Detection System) para realizar a detecção do ataque Slowloris [de Sousa Araújo et al. 2017]. A primeira ferramenta analisada, denominada Suricata, não gera um número adequado de alertas para que um ataque Slowloris seja detectado. Enquanto a segunda ferramenta, denominada SNORT, [Day and Burns 2011, Albin 2011] realiza a detecção do ataque gerando uma avaliação de memória e processamento. Pascoal *et al.* apresentam um módulo de defesa embasado em uma estratégia seletiva denominada SeVen [Pascoal et al. 2017, Dantas et al. 2014]. Este módulo utiliza funções de probabilidade para definir, dentre as novas requisições, quais serão aceitas ou rejeitadas quando o servidor WEB encontra-se saturado. Quando uma nova solicitação chega à aplicação, SeVen

precisa verificar se há disponibilidade no *pool* de conexões do servidor. Caso o mesmo tenha sido esgotado, a estratégia determinará probabilisticamente qual das conexões estabelecidas deverá ser encerrada para que a nova possa ser atendida. Neste caso, conexões de usuários legítimos podem ser encerradas.

Durcekova *et al.* abordam técnicas de detecção de ataque DoS na camada de Aplicação baseadas em assinatura e anomalias [Durcekova et al. 2012]. No caso de técnicas que baseiam-se em assinatura realiza-se um monitoramento de mudanças estatísticas, selecionando primeiramente um parâmetro do tráfego de entrada. A detecção do ataque ocorre se o tráfego inspecionado apresentar características familiares de uma atividade maliciosa. No entanto esta abordagem apresenta limitações, dada a facilidade existente de variação de tráfego de dados. Em se tratando do Slowloris, como seu tráfego de rede é similar ao de conexões legítimas, uma conexão pertencente a um usuário comum poderia ser confundida com uma originada por um atacante. Já a técnica baseada em anomalia identifica um ataque se seu tráfego dados não apresentar um comportamento similar ao de um tráfego especificado como normal a partir de dados de treinamento. Um grande desafio para esta tratativa encontra-se na determinação de todos os tipos de comportamentos normais segundo os dados de treinamento.

Tripathi *et al.* propõem um sistema de detecção de anomalias que mede a Distância de Hellinger entre dois arranjos de probabilidade, obtidos com base em treinamentos e testes [Tripathi et al. 2016]. Na fase de treinamento, é criado um padrão de normalidade com uma distribuição de probabilidade que é constituída de requisições HTTP completas e incompletas. No caso de um fluxo normal, a maior parte das requisições possuem cabeçalho e corpo da mensagem HTTP completos, restando apenas uma minoria que apresenta estas informações de modo incompleto, semelhante ao ataque de DoS HTTP lento. Na fase de testes, uma comparação é feita entre o padrão normal produzido na fase de treinamento e o padrão mais recente de requisições HTTP, utilizando a Distância de Hellinger para determinar a diferença entre os dois arranjos de probabilidade e consequentemente a detecção do ataque de DoS HTTP lento. No entanto, a forma de detecção apresenta falsos positivos relacionados a probabilidade do sistema diferenciar um tráfego malicioso de um tráfego legítimo.

Diferentemente do que é apresentado nos trabalhos relacionados, a ferramenta proposta neste trabalho não limita o número de conexões e também não estabelece *timeout* de conexões. Estes comportamentos podem elevar consideravelmente o número de falsos positivos. Além disso, o presente trabalho realiza o bloqueio do atacante utilizando seu endereço MAC ao invés do IP. Esta ferramenta não depende de análise de processamento e memória do servidor para detectar ataques Slowloris, isto diminui a complexidade da solução proposta. Outra vantagem da ferramenta apresentada é a detecção do tráfego malicioso sem a necessidade de utilizar probabilidade para escolher entre tráfegos legítimos e maliciosos. Conforme demonstrado em trabalhos que utilizaram esta técnica, o número de falsos positivos gerados podem ser elevados.

3. A Ferramenta Proposta

3.1. Análise de Dados

O ataque Slowloris tem como característica a abertura de uma série de conexões incompletas em um servidor WEB. A ideia do ataque consiste em manter as conexões abertas

durante um longo período de tempo e consequentemente esgotar os recursos relacionados a conexão do servidor WEB. Para manter as conexões estabelecidas, diversos pacotes do tipo *Reassembled PDU (Protocol Data Unit)* são enviados em alguns intervalos de tempo. Desta forma, para propor uma solução capaz de identificar este tipo de ataque, é necessária a análise das características comportamentais do ataque Slowloris. Dessa forma, alguns cenários de tráfego foram analisados objetivando obter as referidas características:

3.1.1. Cenários de Análise

Inicialmente, busca-se definir os filtros que serão aplicados sobre as conexões direcionadas ao servidor WEB. O objetivo destes filtros é possibilitar a identificação da incidência de ataque Slowloris. Neste trabalho optou-se por utilizar filtros baseados no comportamento do ataque ao invés de utilizar a limitação de número de conexões. As informações dos *logs* de tráfego foi capturada pela ferramenta Tshark gerando um arquivo de extensão (pcap), e consequentemente visualizada pela ferramenta Wireshark. Dois cenários de tráfego foram analisados e são descritos a seguir:

1. **Analisando a Conexão de Ataque:**

   A análise do ataque foi realizada utilizando como base um cenário de ataque Slowloris padrão, no qual um atacante efetua um ataque direcionado a um servidor WEB. Os *logs* de tráfego foram obtidos utilizando a ferramenta Tshark. O uso do Tshark possibilitou identificar algumas características pertencentes às conexões do atacante demonstrado na Figura 1. Uma particularidade observada é que o primeiro GET realizado na conexão de ataque é um *Reassembled PDU*, com um tamanho de 296 bytes. Os demais GET's, que também são *Reassembled PDU* apresentam um tamanho padrão de 74 bytes. Conforme observado, após estabelecer a conexão com o alvo, o ataque Slowloris envia a primeira solicitação GET de tamanho 296 bytes, já sendo um *Reassembled PDU*. Para manter a conexão aberta, uma série de pacotes *Reassembled PDU* de tamanho reduzido (74 bytes) são enviados periodicamente ao hospedeiro da aplicação com o intuito de reiniciar o *timeout* da conexão. Consequentemente, uma série de conexões deste tipo são abertas pelo ataque, consumindo os recursos de sockets disponíveis no servidor e ocasionando a negação de serviço do servidor.

```
192.168.70.11 → 192.168.70.26 TCP 296 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
192.168.70.26 → 192.168.70.11 TCP 66 80 → 60276 [ACK] Seq=1 Ack=231 Win=30080 Len=0
192.168.70.11 → 192.168.70.26 TCP 296 GET / HTTP/1.1  [TCP segment of a reassembled PDU]
192.168.70.26 → 192.168.70.11 TCP 66 80 → 60280 [ACK] Seq=1 Ack=231 Win=30080 Len=0
```

Figura 1. Logs da ferramenta Tshark analisando um ataque Slowloris

2. **Analisando Conexão WEB Normal e Conexão WEB com Arquivos de Download**

   A outra análise foi realizada considerando o cenário em que apenas conexões legítimas foram estabelecidas com servidor WEB. Esta avaliação foi feita para realizar uma comparação entre conexões normais (geradas por usuários não maliciosos), apresentado na Figura 2, e conexões oriundas do ataque Slowloris. Os

logs demonstram que após o estabelecimento da conexão entre um cliente e o servidor, o primeiro GET não é um *Reassembled PDU*, e apresenta um tamanho superior a 296 bytes. Além disso, a solicitação GET é realizada por meio do protocolo HTTP, possibilitando a identificação da diferença de tráfego quando comparado a uma solicitação de ataque. Isto porque, conforme mencionado anteriormente, a solicitação GET do tráfego de ataque é apresentada no Tshark como protocolo TCP. O campo protocolo localiza-se na quarta coluna das Figuras 1 e 2.

```
192.168.70.11 → 192.168.70.26 HTTP 434 GET /img4.jpeg HTTP/1.1
192.168.70.26 → 192.168.70.11 TCP 1514 HTTP/1.1 200 OK  [TCP segment of a reassembled PDU]
192.168.70.26 → 192.168.70.11 TCP 1514 80 → 58156 [ACK] Seq=2072 Ack=804 Win=260 Len=1448 [reassembled PDU]
192.168.70.26 → 192.168.70.11 TCP 1514 80 → 58156 [ACK] Seq=3520 Ack=804 Win=260 Len=1448 [reassembled PDU]
192.168.70.26 → 192.168.70.11 HTTP 1273 HTTP/1.1 200 OK  (JPEG JFIF image)
```

Figura 2. Logs da ferramenta Tshark analisando uma conexão normal

Observando o comportamento apresentado nos cenários descritos, foi possível detectar características comuns do ataque Slowloris. Com base nessas características, foram desenvolvidos filtros que possibilitam a identificação deste ataque. Os filtros são descritos a seguir.

3.1.2. Filtros Desenvolvidos

A partir da avaliação realizada em cada um dos cenários descritos anteriormente, alguns filtros foram criados visando a identificação do ataque Slowloris nas conexões direcionadas ao servidor WEB. Os filtros criados foram:

- Filtro para Obtenção de GET
- Filtro para Obtenção de Linhas que Possuem *Reassembled PDU*
- Filtro para Obtenção de conexões que tenham protocolo TCP e tamanho do pacote igual a 296 bytes

A classificação de conexões como pertencentes a um ataque Slowloris utiliza os filtros supracitados. Inicialmente, a partir de uma lista contendo os *logs* das conexões direcionadas ao servidor WEB, denominada lista bruta, o filtro de obtenção de GET's é aplicado gerando um novo arquivo de saída, que contém as conexões que são GET's. Em seguida, este novo arquivo de saída serve como base para a aplicação do segundo filtro, que se destina a coletar as conexões que são marcadas como *Reassembled PDU* e adicioná-las a um novo arquivo de texto, cujo conteúdo são conexões que são GET's, marcadas como *Reassembled PDU*. O último filtro atua no segundo arquivo de saída gerado, buscando identificar conexões cujo protocolo utilizado é o TCP e o tamanho do pacote é igual a 296 bytes. Após aplicar o referido filtro, é gerado um último arquivo contendo as conexões que atendem a todos os requisitos pré-definidos, que são as conexões que originalmente possuem GET's, marcadas como *Reassembled PDU*, que tenham tamanho de pacote igual a 296 bytes, e utilizam o protocolo TCP.

3.1.3. Análise do Campo Protocolo

A partir da análise realizada em cada um dos cenários descritos acima observou-se que, após realizar o estabelecimento da conexão, a ferramenta Wireshark não identifica a pri-

meira solicitação ao servidor WEB como um GET. Isto se deve ao fato do Slowloris não finalizar as requisições GET (adicionando um \n ao final de cada solicitação). No caso de uma requisição normal os objetos da página seriam solicitados e após isso a mesma seria finalizada. Desta forma, o campo protocolo das conexões originadas pelo ataque assume o valor TCP (na quarta coluna dos *logs*), diferente do que ocorre com conexões legítimas nas quais este campo assume o valor HTTP.

3.2. Descrição da Ferramenta

A solução proposta neste trabalho apresenta uma ferramenta capaz de identificar e mitigar ataques de negação de serviço realizados pelo Slowloris. Após a identificação de um comportamento malicioso do ataque supracitado, a ferramenta inicia medidas de contenção do tráfego malicioso. O objetivo da ferramenta fundamenta-se na identificação correta de um ataque Slowloris mediante a utilização de um nó concentrador. Uma correta identificação do atacante previne que usuários legítimos sejam prejudicados por medidas restritivas, gerando um menor número de falsos positivos. A ferramenta ilustrada no exemplo de cenário da Figura 3 contém três módulos:

- Módulo de Coleta, que atua no servidor WEB
- Módulo de Análise e Filtragem, que atua em uma máquina denominada, neste trabalho, Concentrador
- Módulo de Bloqueio, que atua nos roteadores

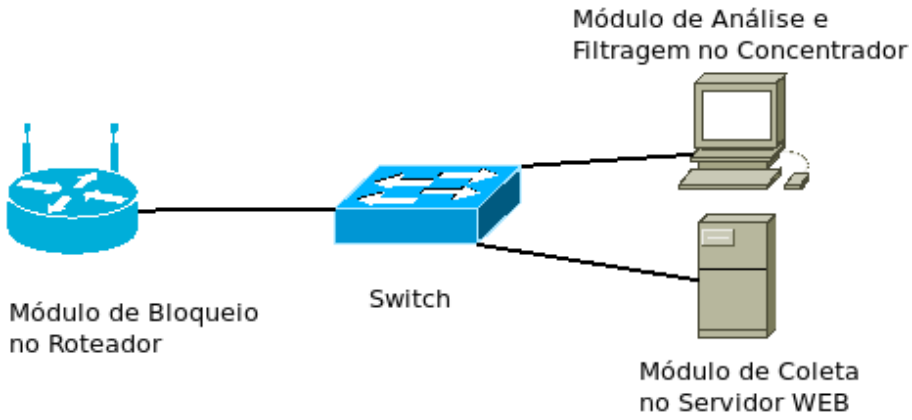

Figura 3. Exemplo de cenário com os módulos sendo apresentados de forma independente

O fluxograma da Figura 4 demonstra o funcionamento do módulo de coleta de dados no servidor WEB. Inicialmente, o módulo do servidor WEB realiza a coleta dos *logs* das conexões a ele direcionadas. Esta coleta é realizada durante trinta segundos usando a ferramenta Tshark, que captura as atividades dos clientes no servidor WEB gerando um arquivo com extensão pcap. Posteriormente é realizada a conversão do arquivo que contém os *logs* das conexões para um arquivo de texto, denominado lista bruta, que será analisado pelo Concentrador.

A Figura 5 mostra o funcionamento do módulo de análise e filtragem presente no Concentrador. A verificação do ataque é feita baseada em filtros previamente estabelecidos, e que serão aplicados na lista bruta em cascata, com o objetivo de identificar a

Figura 4. Módulo de coleta no servidor WEB

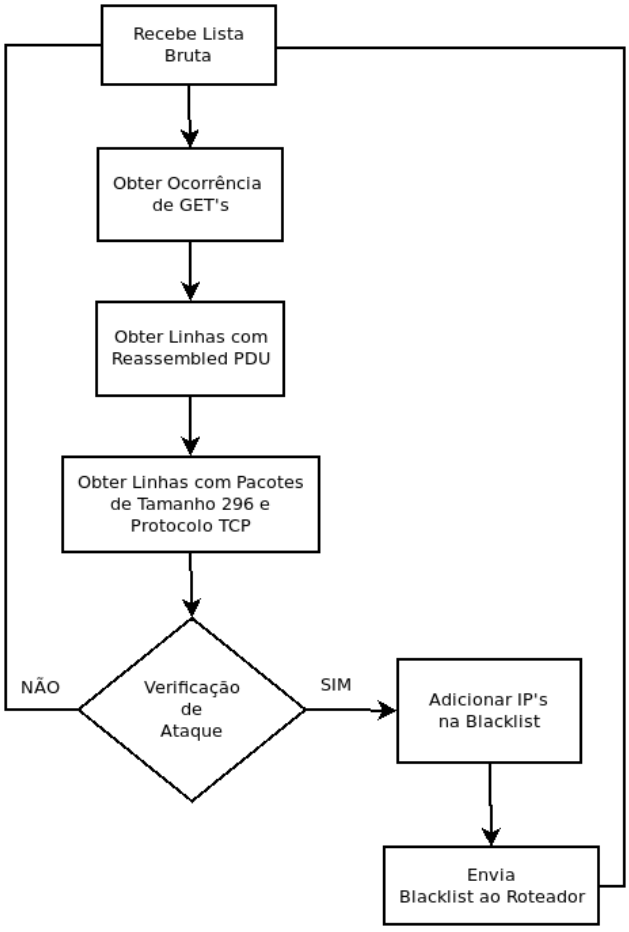

Figura 5. Módulo de análise e filtragem (Concentrador)

ocorrência de ataques. O primeiro filtro obtém todas as conexões nas quais há presença de GET's. O segundo filtro detecta as conexões que são *Reassembled PDU*. O terceiro tem como finalidade realizar duas análises simples, onde a primeira é usada para obter conexões cujo tamanho dos pacotes é de 296 bytes, e a segunda destaca as conexões que têm como protocolo o TCP. Caso haja alguma conexão em que todos esses filtros obte-

nham respostas positivas, assume-se que há ocorrência de ataque e consequentemente a ferramenta obtém o endereço IP de origem dessas conexões. Estes endereços IP serão adicionados em uma lista denominada Blacklist, que será enviada aos roteadores.

O bloqueio do atacante é realizado conforme apresentado na Figura 6. Inicialmente o roteador recebe a Blacklist gerada pelo Concentrador, realizando uma comparação dessa lista com sua tabela ARP para obter os endereços MAC de todos os IP's contidos na Blacklist. Posteriormente, aplica-se regras de bloqueio do IPTABLES aos endereços MAC identificados como atacantes. Após essas etapas, o roteador aguarda por novas Blacklists com o intuito de bloquear novos atacantes. Optou-se pelo bloqueio baseado em endereço MAC, pois gera um menor número de falsos positivos quando comparado com o bloqueio baseado em endereço IP. Evitando com esta abordagem problemas oriundos da alocação dinâmica de endereços IP feita pelo DHCP (Dynamic Host Configuration Protocol). Neste caso, usuários legítimos poderão ser prejudicados se passarem a utilizar o endereço IP identificado como atacante, realizando a negação de serviço para um usuário não malicioso.

Como política de descarte, a ferramenta utiliza como critério a exclusão dos endereços da Blacklist após 72 horas sem incidência de ataque na rede. No entanto, esse tempo pode ser alterado a critério do administrador da rede, que também pode realizar a remoção manual dos endereços maliciosos.

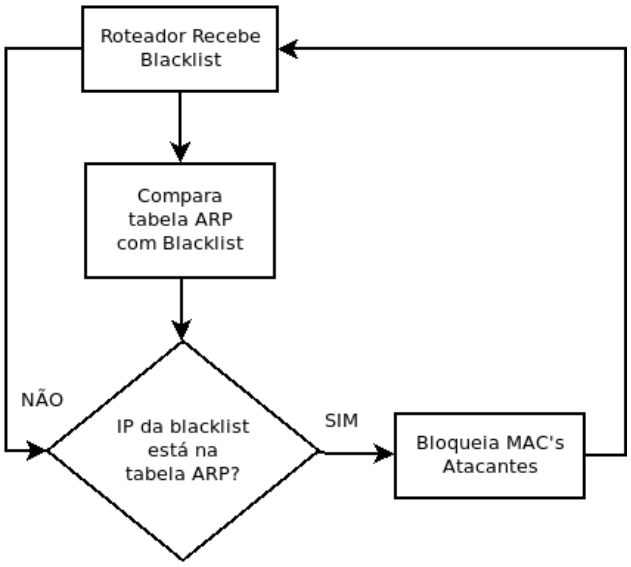

Figura 6. Módulo de bloqueio

4. Resultados

A Figura 7 ilustra o cenário de testes utilizado em um laboratório. Este cenário apresenta uma rede interna na qual os roteadores não fazem uso de NAT (*Network Address Translation*). Os seguintes hardwares foram utilizados: seis Roteadores sem-fio TEW-632BRP, dois Servidores I5-4590 8GB e cinco notebooks AMD A8-4500M com 8 GB de RAM. Os servidores utilizam Ubuntu 16 e os notebooks utilizam Ubuntu 18. Os roteadores tiveram seus firmwares substituídos pelo Open-WRT. Esta substituição foi feita para que fosse

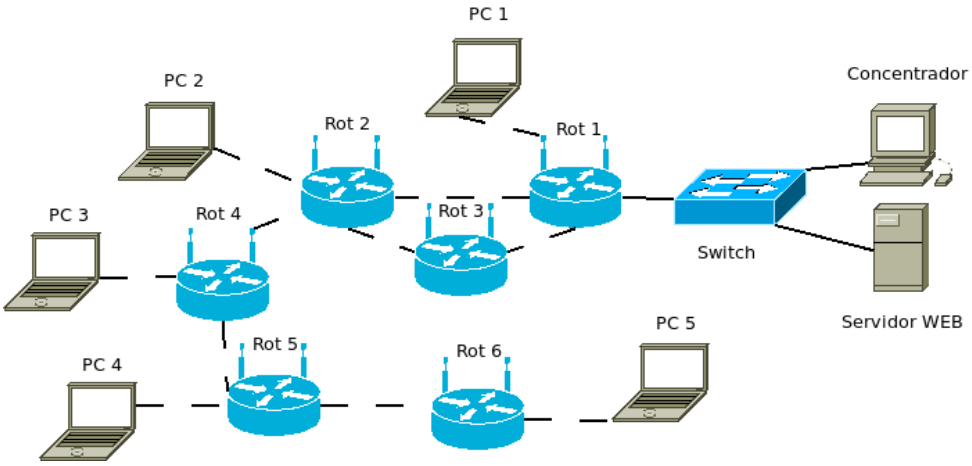

Figura 7. Cenário de testes

possível executar os módulos da ferramenta nos roteadores. Como aplicação, utilizou-se o servidor Apache em sua versão 2.4.7.

Para o cenário apresentado foram coletadas informações referentes a alguns parâmetros da rede visando obter o tempo total de bloqueio. Os parâmetros considerados foram:

- Constante de Coleta (C_C) - Tempo em que o módulo do servidor WEB coleta as informações referentes ao tráfego da rede direcionado à ele, gerando a lista bruta. O tempo estabelecido nos experimentos foi de 30 segundos.
- Tempo de Transferência da Lista Bruta (T_{ELB}) - Representa o tempo necessário para que o módulo de coleta no servidor WEB envie ao Concentrador a lista bruta contendo os *logs* das conexões.
- Tempo de Processamento da Lista Bruta (T_{PLB}) - Representa o tempo necessário para a execução do módulo contido no Concentrador, que originará a Blacklist.
- Tempo de Envio da Blacklist (T_{EBL}) - Representa o tempo necessário para que o Concentrador envie ao roteador a Blacklist contendo os IP's maliciosos. A média encontrada para este parâmetro foi de 0,005 segundos.
- Tempo de Execução do Script de Bloqueio (T_{ESB}) - Representa o tempo necessário para a execução do script de bloqueio em um roteador, a partir da Blacklist recebida do Concentrador. A média encontrada para este parâmetro foi de 0,022 segundos.
- Número de Saltos (NS) - Representa o número de saltos do Concentrador até o roteador em que o atacante está conectado.

A partir dos parâmetros descritos acima, o tempo total de bloqueio (T_{TB}) é especificado a seguir:

$$T_{TB} = (T_{ELB} + T_{EBL}) * 1,1 * NS + C_C + T_{PLB} + T_{ESB} \quad (1)$$

Na equação (1), o valor 1,1 representa a margem de erro de 10% encontrada para o *jitter* da rede nos experimentos. Os parâmetros T_{ELB} e T_{PLB} encontrados nos experimentos podem ser observados na Tabela 1, onde também são apresentados os tamanhos de cada lista bruta utilizada. Os experimentos foram divididos em duas partes:

4.1. Experimento 1

No primeiro cenário, após se conectar à rede, o PC 1, que está a um salto do Servidor WEB, realiza um ataque Slowloris direcionado ao Servidor buscando ocasionar um DoS. O módulo de Coleta que atua no Servidor WEB captura o tráfego de rede direcionado a ele através da ferramenta Tshark, gerando um arquivo de *logs*, denominado lista bruta. Este arquivo é enviado ao Concentrador, que através de seu módulo aplicará todos os filtros preestabelecidos a fim de detectar o endereço IP do atacante. Ao identificar o IP malicioso, ele será adicionado em uma Blacklist e posteriormente enviado ao roteador. De posse da lista de IP's atacantes, o roteador combina esta informação com sua tabela ARP obtendo o respectivo endereço Mac, que será bloqueado no roteador.

Neste experimento variou-se o tamanho da lista bruta com o objetivo de simular diferentes ambientes de redes. Os tamanhos de lista utilizados variaram de 0,646 Mb à 6,22 Mb como mostrado na Tabela 1. Observou-se que neste experimento as mudanças no tempo de detecção e bloqueio do ataque decorrem apenas da variação do tamanho da lista bruta. As mudanças ocorreram nos tempos de Transferência e de Processamento da lista bruta, denominados T_{ELB} e T_{PLB}, e com essa variação o tempo total de bloqueio (T_{TB}) sofreu alterações que podem ser observadas na Figura 8.

Tamanho da Lista Bruta (Mb)	Tempo Transferência (s)	Tempo Processamento (s)
0,646	0,046	0,150
1,82	0,131	0,154
2,8	0,215	0,156
6,22	0.500	0,614

Tabela 1. Tempo médio de transferência e processamento da lista bruta para determinados tamanhos de lista

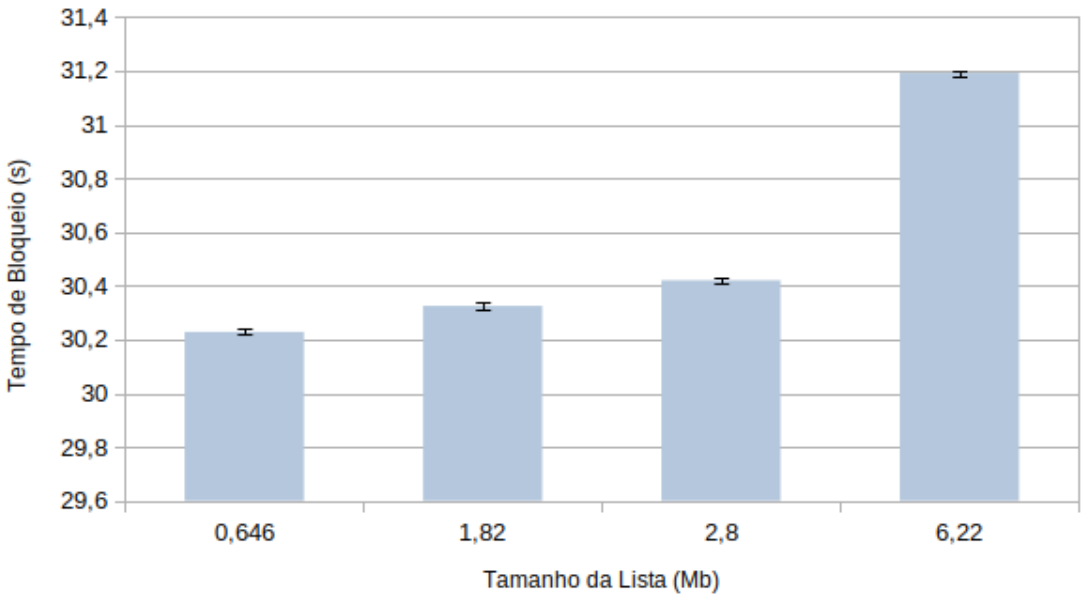

Figura 8. Variação do tempo de bloqueio devido a alteração no tamanho da lista bruta

O experimento foi realizado 5 vezes com o objetivo de obter uma média nos testes. Observou-se que a constante de coleta impactou diretamente o atraso nos gráficos, juntamente com a variação do processamento da lista bruta e o seu consequente envio.

4.2. Experimento 2

No segundo cenário, variou-se o número de saltos do atacante com relação ao Servidor WEB, com o objetivo de calcular o tempo de bloqueio para atacantes que estejam a mais de um salto do alvo. Neste caso, os ataques foram realizados individualmente, ou seja, no primeiro teste o PC 2, que se encontra à dois saltos do alvo realiza o ataque, no teste seguinte o PC 3, que se encontra à três saltos de seu alvo, realiza o ataque, e assim sucessivamente para os demais PC's da rede demonstrados na Figura 7. Os mesmos passos realizados para a detecção e contenção do ataque para o atacante que está a 1 salto (Experimento 1) são os mesmos utilizados para os demais atacantes, que encontram-se a mais de um salto de distância do Servidor WEB. Esses passos se encontram resumidos como se segue:

- Análise do Tráfego
- Geração da Lista Bruta
- Envia Lista Bruta ao Concentrador
- Análise da Lista Bruta
- Criação da Blacklist com Base na Análise da Lista Bruta
- Envio da Blacklist para os Roteadores
- Combinação da Tabela ARP do Roteador com a Blacklist
- Bloqueio dos MAC's

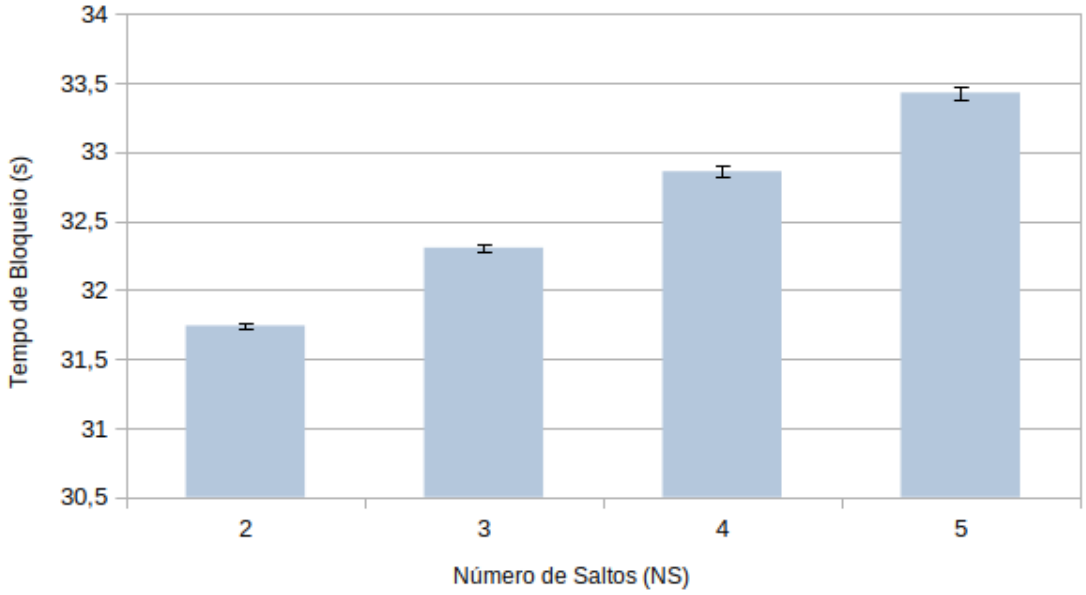

Figura 9. Variação do tempo de bloqueio devido a alteração no número de saltos

Neste experimento variou-se número de saltos entre 2 a 5, como mostrado na Figura 7, mantendo-se constante o tamanho da lista bruta. Observou-se que neste experimento a variação do tempo de detecção e bloqueio do ataque ocorrem devido a variação

do número de saltos. Isto se deve ao fato da Blacklist necessitar de mais tempo para chegar ao roteador no qual o atacante está conectado, devido ao tempo de transferência da mesma entre os roteadores intermediários. Este comportamento pode ser visto no gráfico da Figura 9 onde o tempo de bloqueio aumenta com o número de saltos.

5. Conclusão

Este trabalho apresenta uma ferramenta para detecção e contenção de ataque DoS do tipo Slowloris. A presente proposta realiza uma análise de comportamento do referido ataque objetivando a geração de filtros, que são utilizados para a filtragem dos *logs* das conexões direcionadas ao Servidor WEB. A filtragem dos *logs* proporciona uma correta identificação dos atacantes, permitindo o bloqueio destes e evitando que o ataque interrompa o serviço provido pelo servidor. No primeiro experimento verificou-se que o tempo total de bloqueio varia de acordo com o tamanho da lista bruta. Como resultado, a média do tempo de bloqueio para as listas de tamanho 0,646 Mb e 6,22 Mb é de 30,23s e 31,19s, respectivamente. Estes resultados mostram que a ferramenta consegue mitigar um ataque de Slowloris em andamento em tempo hábil. Uma outra vantagem da ferramenta é a forma como o bloqueio é realizado. Nesta proposta o bloqueio é realizado analisando a conexão com base em filtros preestabelecidos, não limitando o número de conexões permitidas por usuário, diferente do que ocorre em outras abordagens da literatura.

Em trabalhos futuros pretende-se realizar um balanceamento de servidores WEB, com o objetivo de trazer uma maior robustez aos efeitos do ataque. Além disso, busca-se utilizar o servidor Radius atrelado a ferramenta proposta, visando uma identificação mais precisa dos atacantes, tendo em vista que o servidor Radius conta com um banco de dados contendo informações sobre os usuários.

Referências

Albin, E. (2011). A comparative analysis of the snort and suricata intrusion-detection systems. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA.

Barrios Quintanilla, A. and Silva Filho, D. P. d. (2017). Ataques de negação de serviço na camada de aplicação: estudo de ataques lentos ao protocolo http.

Carl, G., Kesidis, G., Brooks, R. R., and Rai, S. (2006). Denial-of-service attack-detection techniques. *IEEE Internet computing*, 10(1):82–89.

Corrêa, A. L. R. and Martins, H. P. (2013). Monitoramento de ataques de negação de serviço: Um caso prático utilizando slowloris. *Faculdade de Tecnologia de Bauru (FATEC)*.

da Silva, C. A. M., Gonçalves, J. A., da Silva Faria, V., de Britto Vieira, G., and Mascarenhas, D. M. (2016). Iremac: Um ips para ataques internos. *XXXIV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*.

Dantas, Y. G., Nigam, V., and Fonseca, I. E. (2014). A selective defense for application layer ddos attacks. In *2014 IEEE Joint Intelligence and Security Informatics Conference*, pages 75–82. IEEE.

Day, D. and Burns, B. (2011). A performance analysis of snort and suricata network intrusion detection and prevention engines. In *Fifth International Conference on Digital Society, Gosier, Guadeloupe*, pages 187–192.

de Sousa Araújo, T. E., Matos, F. M., and Moreira, J. A. (2017). Intrusion detection systems' performance for distributed denial-of-service attack. In *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pages 1–6. IEEE.

Durcekova, V., Schwartz, L., and Shahmehri, N. (2012). Sophisticated denial of service attacks aimed at application layer. In *2012 ELEKTRO*, pages 55–60. IEEE.

Goncalves, J. A., Faria, V. S., Vieira, G. B., Silva, C. A., and Mascarenhas, D. M. (2017). Widip: Wireless distributed ips for ddos attacks. In *2017 1st Cyber Security in Networking Conference (CSNet)*, pages 1–3. IEEE.

Gu, Q. and Liu, P. (2007). Denial of service attacks. *Handbook of Computer Networks: Distributed Networks, Network Planning, Control, Management, and New Trends and Applications*, 3:454–468.

Merino, B. (2013). *Instant traffic analysis with Tshark how-to*. Packt Publishing Ltd.

Orebaugh, A., Ramirez, G., and Beale, J. (2006). *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier.

Papadie, R. and Apostol, I. (2017). Analyzing websites protection mechanisms against ddos attacks. In *2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6. IEEE.

Pascoal, T. A., Correa, J. H., Brayner, R., Nigam, V., and Fonseca, I. E. (2017). Módulo de proteção contra ataques de negação de serviço na camada de aplicação: uma análise de qualidade de serviço e experiência de usuário. In *Anais do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC.

Sangeetha, S. B. Ddos deflate and apf (advanced policy firewall): A report.

Shorey, T., Subbaiah, D., Goyal, A., Sakxena, A., and Mishra, A. K. (2018). Performance comparison and analysis of slowloris, goldeneye and xerxes ddos attack tools. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 318–322. IEEE.

Singh, K., Singh, P., and Kumar, K. (2018). User behavior analytics-based classification of application layer http-get flood attacks. *Journal of Network and Computer Applications*, 112:97–114.

Toklu, S. and Şimşek, M. (2018). Two-layer approach for mixed high-rate and low-rate distributed denial of service (ddos) attack detection and filtering. *Arabian Journal for Science and Engineering*, 43(12):7923–7931.

Tripathi, N. and Hubballi, N. (2018). Slow rate denial of service attacks against http/2 and detection. *Computers & security*, 72:255–272.

Tripathi, N., Hubballi, N., and Singh, Y. (2016). How secure are web servers? an empirical study of slow http dos attacks and detection. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 454–463. IEEE.

Yuan, H., Xia, Y., Yang, H., and Yuan, Y. (2018). Resilient control for wireless networked control systems under dos attack via a hierarchical game. *International Journal of Robust and Nonlinear Control*, 28(15):4604–4623.