Ameaça na Palma da Mão:

Smartphones como Vetores de Ataques Slow de Negação de Serviço

Lucas Oliveira Costa Aversari

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2017

Lucas Oliveira Costa Aversari

Ameaça na Palma da Mão: Smartphones como Vetores de Ataques Slow de Negação de Serviço

Monografia apresentada ao curso de Graduação em Ciência da Computação do Centro de Informática, da Universidade Federal da Paraíba, como requisito para a obtenção do grau de Bacharel em Ciência da Computação

Orientadora: Prof.ª Dr.ª Josilene Aires Moreira

Maio de 2017

Ficha Catalográfica elaborada por
Rogério Ferreira Marques CRB15/690

A952a Aversari, Lucas Oliveira Costa.
Ameaça na palma da mão: smartphones como vetores de ataques slow de negação de serviço / Lucas Oliveira Costa Aversari. – João Pessoa, 2017.
43p. : il.

Monografia (Bacharelado em Ciência da Computação) – Universidade Federal da Paraíba - UFPB.
Orientadora: Profª. Dra. Josilene Aires Moreira.

1. Sistema operacional. 2. Slow DoS. 2. Dispositivos móveis. 3. Segurança Web. I. Título.

UFPB/BSCI CDU: 004.451 (043.2)

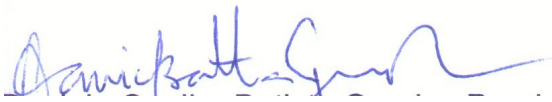
CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Graduação em Ciência da Computação intitulado "Ameaça na Palma da Mão: Smartphones como Vetores de Ataques Slow de Negação de Serviço" de autoria de Lucas Oliveira Costa Aversari, aprovada pela banca examinadora constituída pelos seguintes professores:

Profª. Drª. Josilene Aires Moreira
Centro de Informática – Universidade Federal da Paraíba

Prof. Dr. Raoni Kulesza
Centro de Informática – Universidade Federal da Paraíba

Profª. Drª. Giorgia de Oliveira Mattos
Centro de Informática – Universidade Federal da Paraíba



Coordenador(a) do Curso
Daniela Coelho Batista Guedes Pereira
CI/UFPB

João Pessoa, 5 de junho de 2017

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

“A única maneira de fazer um bom trabalho é amando o que você faz. Se você ainda não encontrou, continue procurando. Não se desespere. Assim como no amor, você saberá quando tiver encontrado.”

Steve Jobs

DEDICATÓRIA

Dedico este trabalho a minha querida família, em especial minha mãe, meu pai e minha irmã que, com muito carinho e apoio, não mediram esforços para que eu alcançasse esta etapa da minha vida. Reconheço todo o empenho de meus familiares e amigos próximos, os quais sempre apoiaram nas mais diversas decisões da vida, fazendo tudo que fosse preciso para que eu alcançasse meus sonhos, por mais distantes que eles parecessem.

AGRADECIMENTOS

Agradeço profundamente a todos que me proveram suporte durante a caminhada ao longo de meu curso, em especial à Prof.ª Dr.ª Josilene Aires Moreira e ao Prof. Dr. Raoni Kulesza, os quais me orientaram na escrita deste trabalho e que, sobretudo, não são apenas meus professores, mas sim grandes amigos que carregarei para o resto da minha vida. Também sou grato por todo o empenho da minha coordenadora de curso e estimada Prof.ª Dr.ª Daniela Coelho Batista Guedes Pereira, que sempre se prontificou a resolver seja lá qual fosse a situação que eu me encontrasse. No mais, agradeço a dedicação dos professores Prof. Hamilton Soares da Silva e Lucídio dos Anjos Formiga Cabral, grandes amigos, diretores de centro e mestres na arte de viver.

RESUMO

Dada a evolução de hardware e software dos dispositivos móveis, a nova geração de smartphones possui grande capacidade de processamento e conectividade. Nesse contexto, tais características, aliadas a sistemas operacionais baseados em Unix (Android e iOS), os tornam potenciais vetores de ataques de negação de serviço, principalmente do tipo *slow*, que utilizam quantidades baixíssimas de largura de banda, processador e memória. Este trabalho apresenta uma ferramenta que pode ser executada em forma de aplicativo para dispositivos Android a qual, implementada exatamente conforme o ataque *slowloris*, é capaz de indisponibilizar um serviço web de médio porte por meio de apenas um dispositivo móvel. A solução proposta foi avaliada e os resultados foram comparados com a versão *desktop* da ferramenta.

Palavras-chave: Slow DoS, Dispositivos móveis, Segurança Web.

ABSTRACT

Given the hardware and software evolution of mobile devices, the new generation of smartphones comes with great processing power and connectivity. In this context, these features, coupled with Unix-based operating systems (Android and iOS), turn them into potential vectors for denial-of-service attacks, especially salow ones, which use low bandwidth, processor and memory. This work presents an tool capable of running as an Android device application, implemented exactly as the *slowloris* attack, being capable of disabling a medium-sized web service by it's execution in a single mobile device. The proposed solution was evaluated and the results were compared with the desktop version of the tool.

Key-words: Slow DoS, Mobile devices, Web security.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

RAM	–	*Random Access Memory* (memória de acesso randômico)
CPU	–	*Central Processing Unit* (unidade de processamento central)
EB	–	Exabytes
DDoS	–	*Distributed Denial of Service* (negação de serviço distribuída)
DoS	–	*Denial of Service* (negação de serviço)
TCP	–	*Transmission Control Protocol* (protocolo de controle de transmissão)
URL	–	*Uniform Resource Locator* (localizador uniforme de recursos)
IIS	–	*Internet Information Services*
TTS	–	*Time to Service* (tempo para serviço)
HTTP	–	*Hyper Text Transfer Protocol* (Protocolo de transferência de hipertexto)

SUMÁRIO

1. INTRODUÇÃO

1.1. Contextualização

Segundo Cambiaso, Papaleo e Aiello (2015), com o advento de novas tecnologias, aparelhos celulares cada vez mais sofisticados são entregues nas mãos dos consumidores a preços acessíveis, fazendo com que a utilização de dispositivos móveis cresça a cada ano e que estes venham, gradativamente, substituindo o uso de dispositivos como *laptops* e *desktops*.

Devido à conectividade embarcada, tais dispositivos representam grande fatia do uso de tráfego na internet. Por exemplo, de acordo com a Cisco (2015), a quantidade de dados trafegados por dispositivos móveis na Internet cresceu 74% em 2015, quando o tráfego de dados móveis atingiu 3,7 exabytes (EB), comparados aos 2,1 EB gerados em 2014. Mais de meio bilhão de novos dispositivos móveis foram conectados à Internet em 2015, sendo o número total de dispositivos conectados 7,9 bilhões, refletindo um aumento de 43% no número médio de usuários de *smartphones* em relação a 2015. Estes *smartphones* são responsáveis por 97% de todo o tráfego de internet gerado pelos dispositivos móveis. A Figura 1 mostra a previsão de crescimento da quantidade de dados gerados mensalmente por dispositivos móveis, estimando um aumento de praticamente 10 vezes até 2020.

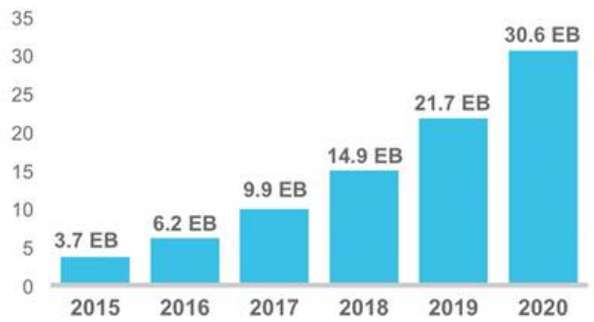


Figura 1. Estimativa da quantidade de dados gerados por dispositivos móveis em Exabytes/mês (EB) até 2020. Fonte: Cisco, 2015.

Por outro lado, além de originar tráfego legítimo, os dispositivos móveis também são fontes de ataques cibernéticos. Tanto Vishrut (2011), como Coursen (2007) ressaltam que o primeiro ataque conhecido a partir de dispositivos móveis data

do ano 2000, quando um *Worm* (tipo de vírus de computador que é auto-replicante em uma rede de computadores) conhecido como Timofonica destinava-se a infectar dispositivos e enviar mensagens de texto a números aleatoriamente gerados.

Entretanto, o cenário geral se alterou desde então: segundo relatório da empresa Symantec, especialista em segurança, entre 2013 e 2015 houve um aumento de 246% (de 127 em 2013 para 528 em 2015) no número de diferentes vulnerabilidades que podem atingir os dispositivos móveis (SYMANTEC, 2016).

Já a empresa Microsoft afirma que um dos tipos de ataques mais utilizados em todo o mundo é o ataque distribuído de negação de serviço, ou DDoS (do inglês, *Distributed Denial of Service*) (MICROSOFT, 2015). A Symantec (2016) afirma que estes ataques são populares por serem simples de se implementar, difíceis de serem mitigados e muito efetivos.

Em um ataque DDoS a partir de computadores, tipicamente o atacante simula requisições simultâneas em grande quantidade para sobrecarregar um determinado servidor e a rota de acesso ao mesmo, deixando-o indisponível para os usuários legítimos. Ataques DDoS, por conta da sua natureza distribuída e colaborativa têm se mostrado um grande problema para os administradores de redes e são considerados atualmente uma das ameaças mais perigosas da Internet (DANTAS; NIGAM; FONSECA, 2014).

Adicionalmente, atacar serviços de grande utilidade pública em períodos de muita utilização dos mesmos como forma de protesto vem se tornando uma tendência. No ano de 2016, devido a atos de terrorismo acontecidos na Europa, um grande ataque DDoS indisponibilizou diversos sites de serviços populares da Internet incluindo o Netflix e o Spotify, além de restringir o acesso a vários sites espalhados pelos Estados Unidos e Europa. O alvo dos ataques foi um conjunto de servidores-mestre DNS (do ingês, *Domain Name Service*), fazendo com que todos os domínios abrangidos por eles se tornassem inacessíveis (THE GUARDIAN, 2016).

1.2. Motivação

Com o crescimento do uso e da representatividade dos dispositivos móveis, especialmente smartphones, os mesmos se tornaram tanto alvos, como potenciais vetores de ataques, sendo explorados pela comunidade hacker para diversos tipos de ações maliciosas, desde o roubo de informações até o prejuízo a serviços de utilidade pública. Particularmente, devido às facilidades oferecidas para sua geração e execução, a categoria de ataques DoS na camada de aplicação é passível de ser disparada através destes dispositivos (CAMBIASO; PAPALEO; AIELLO, 2015).

Nesse contexto, existem dois tipos de ferramentas mais comuns de ataques DoS: *standalone*, onde a execução do ataque pode partir de um único dispositivo, sem a necessidade de um servidor de comando, e *botnet* onde existe um servidor de comando e controle que dispara ataques coordenados a partir de uma rede de dispositivos infectados por software malicioso. Nos últimos anos foram desenvolvidas diversas ferramentas para a execução de tais tipos de ataques a partir de dispositivos móveis, com eficácia comprovada na literatura quando se trata de indisponibilizar serviços Web (CAMBIASO; PAPALEO; AIELLO, 2014).

Pretende-se assim, com este trabalho, apresentar e avaliar a implementação de uma ferramenta de ataque *slow DoS standalone*, voltada para a execução em dispositivos móveis Android, a partir do mesmo *script* em *perl* utilizado na versão *desktop* do *Slowloris*, sem nenhuma modificação para ganho de desempenho. Isto é possível por meio do envelopamento do código original, fazendo uso de bibliotecas do sistema e de uma interface gráfica para simplicidade. Pretende-se assim, verificar a viabilidade da execução de um ataque *slow* DoS escrito em linguagem interpretada a partir de dispositivos móveis Android.

1.3. Objetivo Geral

Como objetivo geral, esta pesquisa visa desenvolver e avaliar uma implementação de uma solução de ataque do tipo *slow* DoS voltada para a execução *standalone* em dispositivos móveis.

1.4. Objetivos Específicos

Os objetivos específicos desta pesquisa podem ser identificados abaixo, em sequência, como:

1. Definir uma arquitetura capaz de suportar a portabilidade de ataques *slowloris* para a execução em dispositivos móveis;
2. Implementar uma aplicação para a execução do ataque DoS *slowloris* em dispositivos móveis utilizando a arquitetura especificada;
3. Planejar a forma de avaliação da solução proposta nos cenários *mobile* e *desktop*;
4. Realizar testes de carga do ataque DoS no cenário e comparar os resultados obtidos pela ferramenta proposta e por sua implementação original para *desktops*.

1.5. Estrutura da Monografia

Este trabalho está dividido em sete capítulos, com os seguintes tópicos: Introdução, onde é apresentada a contextualização do problema e os objetivos gerais e específicos desta monografia; Fundamentação Teórica, a qual apresenta os conceitos relevantes para o entendimento da ferramenta proposta neste trabalho; Trabalhos Relacionados, que mostra trabalhos similares já realizados pela academia e faz uma comparação entre as abordagens já propostas e a presente neste trabalho; Ferramenta Proposta, onde é descrita a implementação e a arquitetura da ferramenta proposta, bem como seus detalhes e módulos; Cenário e Configurações dos Experimentos, que discorre sobre o cenário utilizado nos testes e experimentos, descrevendo seus componentes e configurações; Apresentação e Análise dos Resultados, onde se realiza a apresentação e análise dos resultados obtidos nos testes, além de fazer uma comparação com os resultados obtidos pela versão *desktop* da implementação proposta; e Considerações Finais e Trabalhos Futuros, a qual resume a pesquisa realizada, apresenta suas contribuições e aponta possíveis trabalhos futuros relacionados ao tema.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados, com a finalidade de permitir uma maior compreensão do detalhamento técnico das atividades e resultados expostos nos capítulos seguintes, alguns dos tópicos e conceitos utilizados no decorrer do desenvolvimento deste trabalho.

2.1. Protocolo HTTP

Segundo Zhao (2002) HTTP é um protocolo de transferência de hipertexto baseado em requisição-resposta, no modelo computacional cliente-servidor. É o protocolo utilizado como base na Web para sites e serviços desde 1990, tendo como versão atual o HTTP/1.1. Por esse motivo, optou-se por iniciar este capítulo com uma breve descrição deste protocolo e de suas características, as quais serão importantes para a compreensão dos itens seguintes.

Para a troca de mensagens, o cliente precisa estabelecer uma conexão TCP com uma porta específica do servidor (normalmente a porta 80). O servidor ouvindo naquela porta espera por requisições do cliente. Tal processo de iniciação de conexão seguido por troca de mensagens e finalização é conhecido como sessão HTTP.

O protocolo HTTP define oito métodos básicos para requisições, dos quais quatro se destacam: GET, POST, PUT e DELETE. O método GET requisita uma instância de um determinado recurso ao servidor, recuperando dados, a exemplo de uma imagem ou texto. POST faz o caminho contrário, enviando um conjunto de dados para serem processados pelo recurso especificado, como, por exemplo, no ato de preenchimento de um formulário Web. Informações em um determinado recurso podem ser editadas pelo método PUT ou apagadas por DELETE.

Além disso, parâmetros para controle de tratamento das mensagens de requisição pelo servidor também podem ser disponibilizados em seus cabeçalhos, a exemplo dos campos "*Byte-Range*", "*Context-Length*" e "*Content-Type*", que informam, respectivamente, se o conteúdo é valido (se a posição do *byte* inicial é menor que a do final), tamanho e tipo do conteúdo requisitado e/ou postado.

Dados relevantes e requisitados com frequência em sessões HTTP podem ser armazenados pelo cliente em forma de *cookies* enviados pelo servidor. *Cookies* são arquivos ou strings utilizados como dados pelo navegador ou aplicação, realizando o papel de persistência nas sessões HTTP. São frequentemente utilizados para salvar valores de campos preenchidos, como formulários, nomes de usuário e senhas.

2.2. Ataques *DoS*

Segundo Dantas, Nigam e Fonseca (2014), ataques que fazem o uso de vulnerabilidades ou congestionamento da camada de rede se mostraram grandes ameaças desde os primórdios da Internet, preocupando empresas e gestores de serviços de grande importância para a sociedade.

Ataques que realizam o congestionamento da camada de rede são conhecidos como *flooding* DoS. Tal modalidade de ataque é, normalmente, realizada pelo recrutamento de "zumbis" por meio de software malicioso. Tais máquinas ou equipamentos infectados realizam o envio de grandes quantidades de requisições (pacotes), de maneira coordenada e sincronizada, para um servidor alvo, inundando a fila de sua interface de rede ou esgotando os seus recursos.

Ataques de *flooding* também podem ser executados de maneira pontual, por uma única máquina e por parte de seus usuários, possuindo impacto apenas em serviços de porte mínimo.

Outro fato relevante sobre os ataques de *flooding* é seu alto consumo de recursos da máquina atacante e sua fácil detecção por soluções de monitoramento e gerência de rede, resultando em métodos conhecidos de mitigação, como o bloqueio dos IPs atacantes ou o reencaminhamento de requisições. (DANTAS, 2015)

2.3. Ataques *slow DoS*

De acordo com Cambiaso, Papaleo e Aiello (2014), o fenômeno dos ataques *slow DoS* representa uma evolução recente dos ataques *flooding* DoS, e correspondem a um conjunto de ameaças consolidadas e de simples execução que representam um perigo constante aos administradores de serviços Web.

Diferentemente da primeira geração de ataques *DoS,* que estava focada na inundação das camadas de rede e de aplicação pelo envio de um número exorbitante de requisições (*flooding*) que faziam com que todos os serviços do alvo e da rede ficassem indisponíveis, os ataques do tipo *slow* DoS fazem uso de largura de banda extremamente baixa para tornar um determinado serviço Web indisponível. Para realizar tal feito, tais ataques trabalham explorando protocolos da camada de aplicação do modelo ISO/OSI, afetando diretamente a porta do serviço desejado. (DANTAS; NIGAM; FONSECA, 2014)

2.3.1. *Slowloris* (HTTP GET)

Segundo Dantas (2015), na atualidade, a maioria dos ataques é dirigida à camada de aplicação, com pouca geração de tráfego, também conhecidos como *Slow* DoS. Mais precisamente, são aqueles que exploram os métodos HTTP GET/POST, tendo como o principal representante na atualidade o *Slowloris*, implementado por Robert Hansen.

O *Slowloris* (HTTP GET) é um ataque realizado através do protocolo HTTP, explorando suas características e vulnerabilidades. O principal foco é o aproveitamento da característica de *timeout* de um servidor web, que é o tempo máximo em segundos que uma requisição permanece no *buffer* de aplicação até ser atendida. Segundo Cambiaso, Papaleo e Aiello (2015) e Karim, Shah e Salleh (2015), tal *timeout* pode ser descoberto por meio de um *sniffer* de rede por meio de tentativa e erro, mas não é uma tarefa trivial.

O ataque usando *Slowloris* é de difícil detecção por parte dos sistemas *anti-DoS*, pois consegue gerar tráfego a uma taxa baixa e de pequeno volume. O *Slowloris* tenta manter muitas conexões com o servidor web alvo pelo maior tempo possível (RADWARE, 2016). O mesmo inicia solicitando várias conexões TCP ao servidor, onde, em cada uma delas, o atacante envia uma requisição GET incompleta. Na etapa seguinte, em algum instante próximo do *timeout*, um novo cabeçalho incompleto é enviado para manter a conexão ativa (Figura 2).

Um detalhe importante é que uma aplicação web não pode responder a uma requisição até que ela seja recebida por completo. Com a continuidade do ataque, o número de conexões tende a aumentar, consumindo assim todos os recursos da fila de atendimento da aplicação, levando o servidor a um estado de indisponibilidade, não conseguindo com isso responder às próximas requisições (DANTAS, 2015).

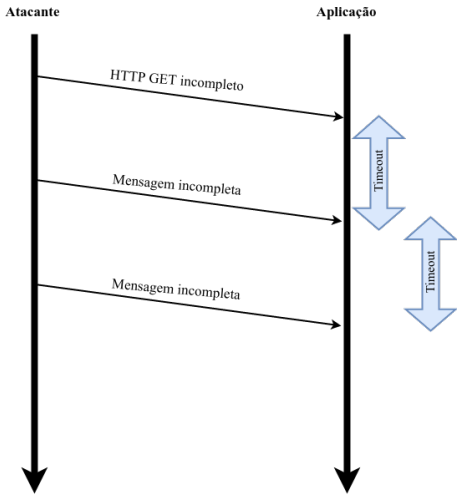


Figura 2. Representação do funcionamento do *Slowloris*

Atualmente, segundo Dantas, Nigam e Fonseca (2014, p. 1) ataques do tipo *slow* DoS, estão extremamente em voga, sendo a ferramenta de ataque *Slowloris* uma das ameaças mais populares. Isso a torna uma das maiores preocupações dos administradores de rede e mantenedores de serviços de grande porte.

No cenário de um ataque s*low* DoS colaborativo utilizando dispositivos móveis ou fixos, todos os dispositivos executando a ferramenta de ataque têm como alvo indisponibilizar um serviço hospedado num determinado domínio (URL) geralmente um servidor web (Figura 3). A indisponibilidade tende a se manter enquanto durar o ataque (KARIM; SALLEH; SHAH, 2015).

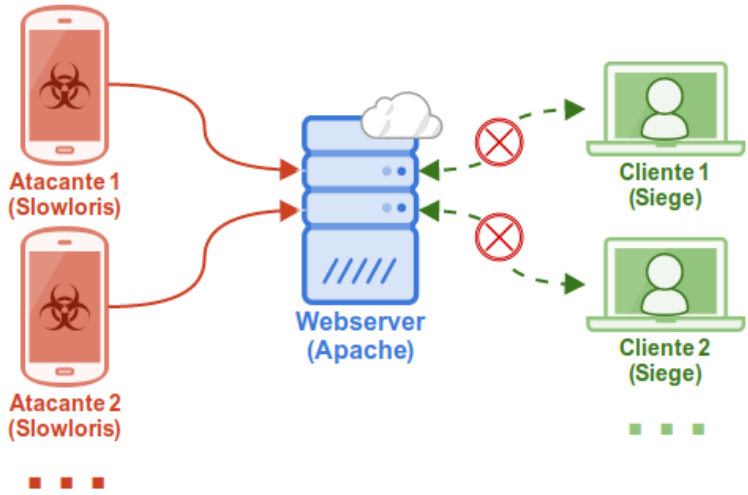


Figura 3. Cenário padrão de um ataque Slow DoS mobile

2.4. Perl Script

Perl (2017) é uma linguagem de programação multiplataforma usada principalmente em aplicações Web, para administração de sistemas linux e por aplicações que realizam intenso processamento de strings. A linguagem foi desenvolvida por Larry Wall, em 1987, para facilitar o processamento de texto e formulários, substituindo diversas extensões e plug-ins necessários.

A sintaxe da linguagem é extremamente simples e parecida com a da linguagem C, contendo variáveis, expressões e etc. No entanto, pequenos códigos em *perl* podem realizar uma grande quantidade de ações, devido às chamadas ao sistema operacional.

Perl possui diversos módulos de bancos de dados e suporte a HTTP, permitindo, também, o acesso a bibliotecas externas em outras linguagens, como C/C++. Novas funcionalidades podem ser adicionadas por meio de extensões, amplamente disponibilizadas no arquivo da linguagem.

2.5. Android

Android (2017) é um sistema operacional móvel baseado no núcleo linux, desenvolvido pelo Google. Sendo o sistema operacional móvel mais utilizado do

mundo, representando 80% da fatia de mercado, devido a seu código fonte aberto e disponibilizado.

Foi apresentado em 2007, possuindo versão comercial apenas em 2009. Atualmente, o Android conta com diversos recursos de usabilidade e dispõe de basicamente todas as funcionalidades presentes em uma distribuição linux padrão, possibilitando a execução de diversos tipos de aplicações.

Segundo Brähler (2010), a arquitetura do Android é organizada em modelo de pilha, tendo o núcleo linux na camada mais baixa e o topo formado pelas aplicações em execução, sendo algumas delas partes essenciais do sistema, como o discador telefônico.

Desenvolvedores possuem acesso às mesmas bibliotecas que o sistema disponibiliza para aplicações internas, simplificando o reúso de componentes e facilitando o processo de desenvolvimento. O Android inclui um vasto conjunto de bibliotecas de C/C++, Java, Python e outras linguagens que são expostas para os desenvolvedores por meio do *framework* de aplicação.

Novas bibliotecas podem ser integradas ao *framework* de aplicação, permitindo que o suporte a novas ferramentas e linguagens seja adicionado de maneira simples, fácil e portável, pois os métodos de bibliotecas externas ao *framework* de aplicação residem diretamente no pacotes dos aplicativos que fazem uso das mesmas. Toda essa integração é feita por meio do ANDK (do inglês, *Android Native Development Kit*), que são interfaces para a chamada de métodos e bibliotecas escritas em diversas linguagens, a exemplo de Java e a JNI (do inglês, *Java Native Interface*). (BRÄHLER, 2010)

3. TRABALHOS RELACIONADOS

Neste capítulo, reportamos parte dos trabalhos relacionados ao tópico de ameaças a partir de dispositivos móveis, cobrindo desde soluções que englobam outros tipos de ataques até um cenário mais específico, com ataques de negação de serviço do tipo *slow*.

O panorama atual provém dos trabalhos de Cambiaso, Papaleo e Aiello (2015) e de Karin, Shah e Salleh (2015), que, além de implementarem soluções de ataques a partir de dispositivos móveis como forma de estudos de caso, realizaram uma taxonomia deste tipo de ameaças, sendo as mais relevantes destacadas nas seções seguintes.

Além disso, parte relevante do histórico dos ataques *slow* DoS a partir de computadores também é destacada neste capítulo, sendo importante para a compreensão da sua evolução para os dispositivos móveis.

3.1. Ataques *Slow DoS Desktop*

Entre os primeiros ataques *slow* DoS documentados, destaca-se o denominado de *Apache Range Header*, publicado na Internet como um *script* feito pelo usuário denominado "KingCope". Tal ataque explora o parâmetro "byte-range" do protocolo HTTP para forçar o servidor Web a fazer réplicas em memória de uma determinada requisição. (ALAM, 2014);

Depois do "KingCope", várias outras ferramentas de ataque surgiram fora da comunidade científica e se espalharam livremente pela Internet, ganhando popularidade devido ao seu alto impacto oferecido. Entre estes ataques, destaca-se o *Slowloris*, uma das ferramentas *slow* mais conhecidas e utilizadas, atingindo o alvo com uma grande quantidade de requisições HTTP pendentes (RADWARE, 2016).

Em 2013, Sergey Shekyan lançou uma ferramenta denominada de *slowhttptest*, consistindo de uma compilação de ataques *slow DoS* conhecidos, junto com um novo ataque desenvolvido pelo time de Sergey, o *slowread*, que simulava clientes lentos para congestionar o servidor Web (SLOWHTTPTEST, 2017).

3.2. Ferramentas de Ataque Móveis

Atualmente, uma das ameaças móveis amplamente populares na Internet é denominada de *WiFiKill*, que é uma aplicação projetada para desabilitar o acesso de determinados dispositivos a uma rede WiFi compartilhada em que o atacante esteja conectado, causando transtornos aos usuários, principalmente em ambientes públicos (WIFIKILL, 2014).

Outro ataque que tem como vetor um dispositivo móvel é denominado *DroidSheep*, consistindo de um aplicativo que dá ao atacante habilidades para coletar *cookies* em uma rede compartilhada, permitindo, assim, que dados de sessões de usuários sejam desviados, com o intuito de roubar informações das vítimas (DROIDSHEEP, 2017). Várias outras ferramentas a mesma proposta encontram-se disponíveis, como o *Faceniff* (2017).

3.2.1. Ferramentas de Ataque *DDoS*

Considerando o cenário de ataques de negação de serviço a partir de dispositivos móveis, podem ser encontradas várias aplicações baseadas na ferramenta desktop *LOIC* (do inglês, *low-orbit Ion Cannon*) (2017), desenvolvida pelo grupo de hacktivistas *Anonymous*, capazes de realizar ataques DDoS do tipo *flooding*.

No entanto, ataques de inundação não se mostram adequados e nem escaláveis ao cenário dos dispositivos móveis, devido à grande quantidade de banda e recursos computacionais por eles consumida. Sendo mais adequados para este contexto, os ataques *slow* DoS.

3.2.2. Ferramentas de Ataque *slow DoS*

Tratando-se de ataques *slow* DoS partindo de dispositivos móveis, destacam-se algumas ferramentas, como o *SlowBot* (KARIM; SHAH; SALLEH, 2015), um

cliente de *botnet* onde o atacante encontra-se subordinado a um servidor de comando e controle para realizar ataques que exploram o protocolo HTTP.

Na modalidade de ataques similares ao *slowloris*, tanto o *NeoLoris* (CAMBIASO; PAPALEO; AIELLO; 2014) como o *SlowDroid* (CAMBIASO; PAPALEO; AIELLO; 2015) possuem destaque, caracterizando-se como ataques *standalone* implementados para atingir o protocolo HTTP da mesma forma que a implementação original do *slowloris*. No entanto, ambos os ataques realizam monitoramento do tráfego e de recursos para realizar alocação otimizada para dispositivos móveis.

3.3. Análise Comparativa

Conforme a Tabela 1, diferentemente das ferramentas *SlowBot*, *NeoLoris* e *SlowDroid*, a solução proposta nesta monografia visa realizar a execução do ataque *slowloris*, conforme sua especificação original, em dispositivos móveis Android, sem nenhuma otimização de performance ou rede. Além disso, a ferramenta busca a execução unificada do ataque, de modo *standalone*, possibilitando que um único dispositivo tenha a capacidade de ser vetor de um ataque *slow* DoS.

Outra diferença da solução presente neste texto é que sua execução se dá em linguagem interpretada *perl*, de maneira idêntica à da implementação oficial do *slowloris*, por meio do encapsulamento e uso de bibliotecas de sistema para a execução e passagem de parâmetros.

As outras ferramentas citadas na presente seção não se encontram na tabela comparativa por não se tratarem de ataques *slow* DoS, na camada de aplicação. Os parâmetros de comparação foram escolhidos de forma que as principais diferenças entre as soluções e a proposta fossem destacadas.

Tabela 1. Análise comparativa entre as soluções

Nome da ferramenta	Tipo de ataque	Modo de ataque	Tipo de implementação
SlowBot	Slow	Botnet	Compilada
NeoLoris	Slow	Standalone	Compilada
SlowDroid	Slow	Standalone	Compilada
Solução Proposta	Slow	Standalone	Interpretada e inalterada

4. FERRAMENTA PROPOSTA

A ferramenta de ataque proposta e desenvolvida é uma versão do utilitário *Slowloris*, que utiliza o mesmo *script* da versão desktop para a execução em dispositivos móveis Android. O código em *perlscript* da versão original foi encapsulado pela biblioteca *perlDroid* (2016) e conta com interface gráfica para simplificar o manuseio. Todos os componentes são empacotados dentro do arquivo de aplicação (.apk), podendo ser instalado e executado em dispositivos Android acima da versão 2.3.

A interface projetada para o aplicativo foi inspirada no *NeoLoris,* e contém apenas campos para inserção da URL a ser atacada e seleção da intensidade do ataque (Figura 4), além de um campo que informa o verbo HTTP utilizado (HTTP GET). O papel da interface é a passagem de parâmetros para o módulo da biblioteca *perlDroid*, que por sua vez, executa o *perlscript* original da versão desktop: *slowloris.pl*, desenvolvido, segundo a Radware (2016), pelo usuário anônimo Sergey Shekyan até o ano de 2013.

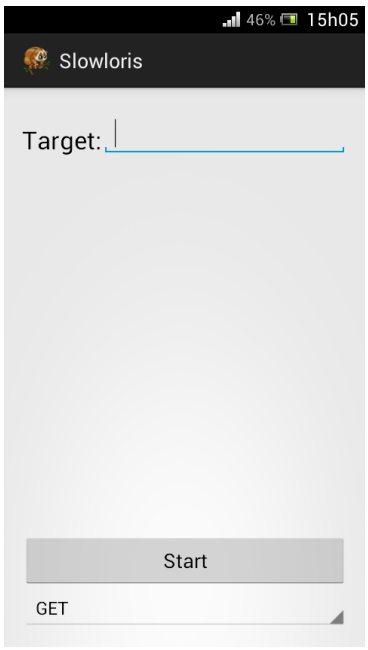


Figura 4. Interface do aplicativo "*Slowloris*" desenvolvido

São usadas quatro diferentes configurações do ataque nos experimentos: *lite, normal, extreme ou uber*, conforme a Tabela 2. Os atacantes são as *threads* responsáveis pelos envios de requisições HTTP GET para o alvo, com um *timeout* definido entre as mesmas. Tal intervalo deriva do fato de o *timeout* HTTP padrão da maioria dos servidores Web disponíveis no mercado ser de 40s.

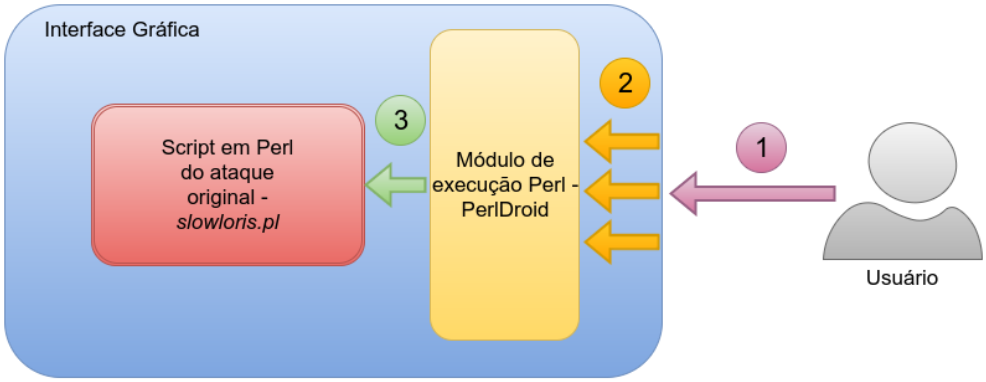
Tais configurações de ataque foram definidas baseadas no fato de que a solução desenvolvida neste trabalho busca atingir servidores de médio porte, que, segundo a Netcraft (2016), são aqueles que possuem capacidade para atender requisições de 750 usuários simultaneamente.

As intensidades de ataque definidas na Tabela 2 permitem que haja, respectivamente, a ocupação de 33%, 66%, 100% e 133% do *buffer* de atendimento do servidor. Caracterizando, no último caso, uma situação extrema.

Tabela 2. Números de atacantes e timeouts correspondentes a cada uma das intensidades de ataque.

Intensidade do ataque	N° de atacantes	Timeout entre requisições
Lite	250	35s
Normal	500	35s
Extreme	750	35s
Uber	1000	35s

A interface da aplicação desenvolvida envia os parâmetros do ataque para o componente *perlDroid* presente no aplicativo, que, por sua vez, chama a versão original do *Slowloris* com os parâmetros escolhidos e a envia para ser executada pelo Android *shell*, a cargo do sistema operacional. O motivo para a predeterminação das quantidades de atacantes é manter a simplicidade, o que implica na facilidade com que os ataques podem ser configurados e iniciados (Figura 5).



Onde:	
1	Usuário seleciona parâmetros do ataque via interface gráfica
2	Interface passa os parâmetros para o módulo PerlDroid
3	Script original é chamado com os parâmetros escolhidos

Figura 5. Funcionamento da aplicação

A ferramenta apresentada se destaca das demais ferramentas de ataques *Slow* DoS *mobile* presentes na literatura pelo fato de ser *standalone* e não possuir nenhuma otimização ou mudança no *script* original do ataque, sendo responsável pela execução do famigerado ataque *Slowloris* a partir de dispositivos móveis, com hardware mais limitado do que o para o que o ataque foi inicialmente projetado.

Na ferramenta proposta cada usuário pode, voluntariamente, realizar um ataque de forma independente, o que constitui um cenário de ataque *slow* DoS, que pode ser distribuído e/ou colaborativo. Esta classe de ferramentas vem se popularizando cada vez mais pela ação nas redes sociais de grupos de hackers como o *Anonymous*, o qual promove a realização de ataques por meio da colaboração dos seus seguidores, que instalam e utilizam voluntariamente aplicativos maliciosos disponibilizados abertamente para realizar ataques a determinadas URLs divulgadas.

5. CENÁRIO E CONFIGURAÇÕES DOS EXPERIMENTOS

Para demonstrar a efetividade da ferramenta de ataque proposta foram realizados experimentos de impacto em um cenário composto por um servidor web, dispositivos executando a ferramenta de ataque desenvolvida e um computador executando a ferramenta de simulação de clientes honestos *Siege*.

O cenário é similar aos utilizados por Dantas (2015) e Cambiaso, Papaleo e Aiello (2015) para simular ataques no desenvolvimento de suas ferramentas, no entanto, foi realizado um aumento para uma escala duas vezes maior, partindo de servidores Web de pequeno porte, para versões de médio porte, conforme a Netcraft (2016).

A escolha do servidor Web usado justifica-se através de dados apresentados pela Netcraft, que mostram que aproximadamente 46% dos sites da Internet são hospedados em servidores Apache 2, representando uma parcela muito superior àquela ocupada por seus concorrentes Nginx e IIS (*Internet Information Services*) (NETCRAFT, 2016).

Assim, utiliza-se o servidor Web Apache 2, utilizando o *mpm_prefork* para processar requisições, com o parâmetro *MaxRequestWorkers* configurado com o valor 750. Ou seja, um servidor de médio porte, capaz de atender a 750 clientes simultaneamente (NETCRAFT, 2016). O *timeout* HTTP do servidor foi mantido na sua configuração padrão de instalação, de 40 segundos (Figura 6).

```
StartServers          5
MinSpareServers       5
MaxSpareServers       10
MaxRequestWorkers     750
ServerLimit           750
MaxRequestsPerChild   250
```

Figura 6. Valores presentes no arquivo de configuração mpm_prefork.conf do Apache.

Para simulação dos clientes honestos, o software *Siege HTTP load test* foi escolhido, como utilitário de *benchmark* e carga no servidor, devido a sua grande presença e referência na literatura (FULMER, 2014). Em todos os testes, 250 clientes honestos foram simulados pelo *Siege* como usuários constantes do servidor web, envian-

do requisições do tipo HTTP GET, mantendo uma taxa de ocupação legítima de aproximadamente 33% do *buffer* de atendimento do servidor, conforme dados da Netcraft (2016), que afirmam que esta é a taxa de ocupação em casos de uso normais.

As métricas coletadas no servidor são a disponibilidade média do domínio, ou seja, qual a taxa de clientes honestos disparados que consegue efetivamente acessar o site e o TTS (*time-to-service*) médio, que é o tempo de serviço médio de tais requisições, ou seja, o intervalo até que elas sejam atendidas pelo servidor. Assim, é possível analisar o impacto na qualidade de serviço dos clientes (DANTAS; NIGAM; FONSECA, 2014). A Figura 7 mostra o cenário dos testes.

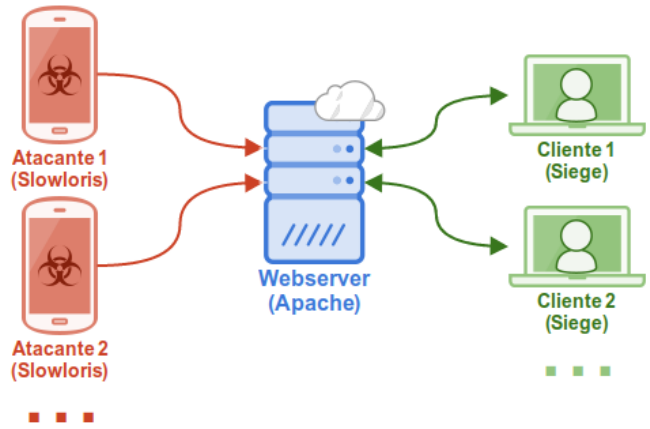


Figura 7: Cenário dos experimentos

Durante a realização dos testes, foram variados os seguintes parâmetros:

- Intensidade do ataque (*lite*, *normal*, *extreme* ou *uber*)
- Número de dispositivos (1 ou 2)

Comparações diretas com os resultados obtidos na versão móvel foram realizadas utilizando os mesmos parâmetros de configuração em chamadas ao *slowloris.pl* na versão desktop, demonstrando a efetividade da versão móvel no cenário de testes proposto. O hardware utilizado nos testes foi:

- Atacantes: Celulares Sony Xperia J (1.00 Ghz, 512 Mb de RAM, Android 4.1.2);
- Webserver: HP PC 5850 (Athlon 64 X2 @ 3.00 Ghz, 4 Gb de RAM, Ubuntu 16.10);
- Clientes: MacBook [late 2009] (Core2Duo @ 2.20 Ghz, 6 Gb de RAM, OS X 10.12 Sierra);

Todos os testes de ataque tiveram duração capturada de 120 minutos e foram acompanhados visualmente, em busca da ocorrência de falhas ou anomalias. A forma de conectividade utilizada tanto nos celulares como no computador que simulava clientes legítimos foi via rede *wireless* isolada, padrão N 300 Mbps. Cada experimento foi realizado trinta e três vezes (33) e então obtida a média de cada uma das métricas, resultando num nível de confiança estatístico de 95%.

Adicionalmente, mais duas métricas foram coletadas durante os experimentos: consumo de CPU e utilização de memória usados pelas ferramentas de ataque e pelo servidor web Apache. Para tanto foram escritos *scripts* para capturar a saída do comando de terminal *htop*. No caso dos dispositivos Android, o *script* foi executado por terminal via o aplicativo de super-usuário *Terminal Emulator*.

6. APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

6.1. Efetividade dos Ataques

6.1.1. Ataques Utilizando Um Dispositivo Móvel

A Tabela 3 apresenta os resultados de disponibilidade e TTS utilizando-se apenas um aparelho celular para a execução dos ataques. Para fins de comparação são mostrados os resultados para a versão *Slowloris* desktop, utilizando-se os mesmos parâmetros de configuração.

Tabela 3: Testes com apenas um dispositivo atacante

***Slowloris* Mobile**			***Slowloris* Desktop**		
Intensidade do Ataque	Disponibilidade (%)	TTS (s)	**Intensidade do Ataque**	Disponibilidade (%)	TTS (s)
Lite	100,00	0,08	**Lite**	100,00	0,06
Normal	98,97	0,29	**Normal**	99,68	0,23
Extreme	0,00	∞	**Extreme**	0,00	∞
Uber	0,00	∞	**Uber**	0,00	∞

Nota-se a consistência dos resultados da versão móvel quando comparada à versão tradicional. Isso se deve ao fato do mesmo código estar sendo executado de maneira encapsulada em outra plataforma que, apesar de possuir funcionalidades limitadas, consegue suprir todas as demandas do *script*. As mínimas variações nos resultados se devem às atividades executadas pelo *webserver* no momento dos ataques.

Também percebe-se que, tanto para a versão mobile como para a versão *desktop*, o servidor só se torna indisponível a partir de ataques cuja intensidade definida é “Extreme”, já que o número de atacantes é igual ao número máximo de clientes que podem ser atendidos simultaneamente pelo servidor negando, assim, serviço a clientes legítimos.

Outro fato importante é o impacto no tempo de serviço (TTS) gerado pelo ataque a partir do momento em que todas as posições do servidor encontram-se ocu-

padas. Isso ocorre no cenário descrito a partir de ataques “Normal”, onde temos 500 atacantes e 250 clientes usufruindo de um *buffer* de atendimento de 750 posições no servidor. Ocorre um aumento no TTS devido ao fato de que novas requisições feitas pelos clientes tem que esperar para serem atendidas, já que todas as posições disponíveis encontram-se em uso.

6.1.2. Ataques Utilizando Dois Dispositivos Móveis

Também foram coletados dados de disponibilidade e TTS em testes realizados utilizando dois aparelhos celulares (Tabela 4), executando o ataque nas mais diversas intensidades (cada um dos dispositivos com a intensidade demarcada abaixo). Como comparação, encontram-se os resultados da execução de duas instâncias equivalentes da ferramenta *Slowloris* na versão *desktop*.

Tabela 4: Testes com dois dispositivos atacantes

***Slowloris* Mobile**			***Slowloris* Desktop**		
Intensidade do Ataque	Disponibilidade (%)	TTS (s)	**Intensidade do Ataque**	Disponibilidade (%)	TTS (s)
Lite	99,48	0,10	**Lite**	99,98	0,11
Normal	0,00	∞	**Normal**	0,00	∞
Extreme	0,00	∞	**Extreme**	0,00	∞
Uber	0,00	∞	**Uber**	0,00	∞

Percebe-se que, neste cenário, a efetividade do ataque é bem maior do que utilizando apenas um dispositivo. Quando dois dispositivos executando o ataque na intensidade “Normal” (500 atacantes) de forma colaborativa atingem o servidor, o mesmo já se torna indisponível, pois só possui capacidade para atender 750 clientes simultaneamente. A partir dos resultados acima é possível perceber que existe consistência no ataque, principalmente quando realizado de forma colaborativa e distribuída.

6.2. Uso de CPU e Memória Pela Ferramenta de Ataque

Serão expostos abaixo os dados da utilização de memória e CPU pela ferramenta de ataque móvel quando configurada em cada uma das intensidades de ataque. Percebe-se um uso muito baixo de memória, mantendo-se abaixo dos 10 Mb em todos os casos.

6.2.1. Intensidade “Lite”

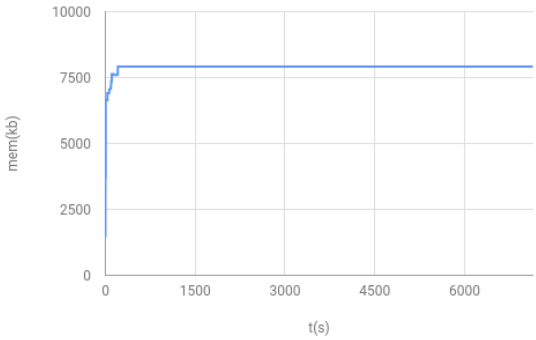


Figura 8. Uso de memória pela ferramenta de ataque na intensidade “Lite”.

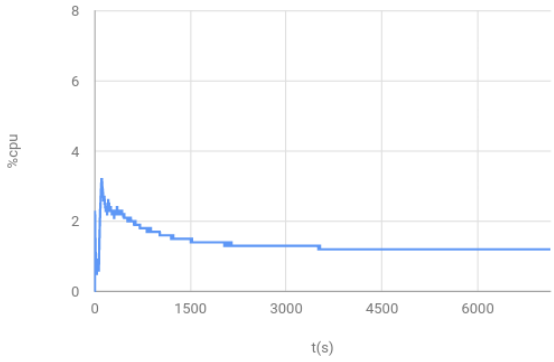


Figura 9. Uso de CPU pela ferramenta de ataque na intensidade “Lite”.

Percebe-se que o uso de memória fica em torno dos 7,75 Mb, enquanto que o uso de CPU não superou os 3% (figuras 8 e 9). O baixo uso de memória e CPU, tanto na versão desktop, como na versão móvel proposta, se deve à baixa complexidade das *threads* atacantes criadas, as quais realizam a simples atividade de enviar um HTTP GET periodicamente.

6.2.2. Intensidade “Normal”

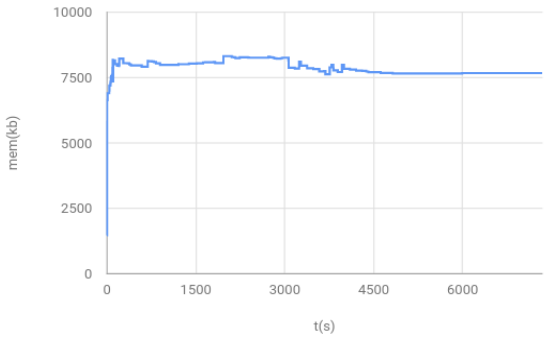


Figura 10. Uso de memória pela ferramenta de ataque na intensidade “Normal”.

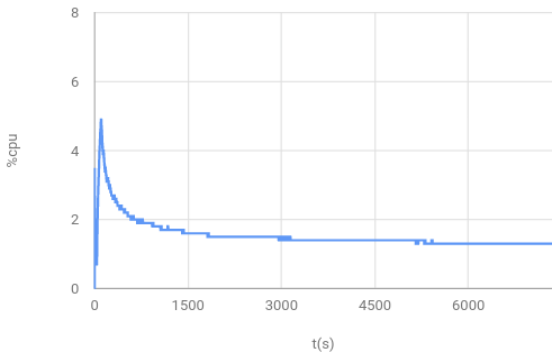


Figura 11. Uso de CPU pela ferramenta de ataque na intensidade “Normal”.

Note que o uso de memória permanece próximo ao valor da intensidade “Lite”, isso se deve ao fato do Android tentar otimizar ao máximo a área de *heap* das *threads* iguais, identificando áreas compartilhadas por elas. Ocorre um pequeno aumento no pico de CPU no momento da inicialização do ataque devido à maior quantidade de *threads* criadas (figuras 10 e 11).

6.2.3. Intensidade “Extreme”

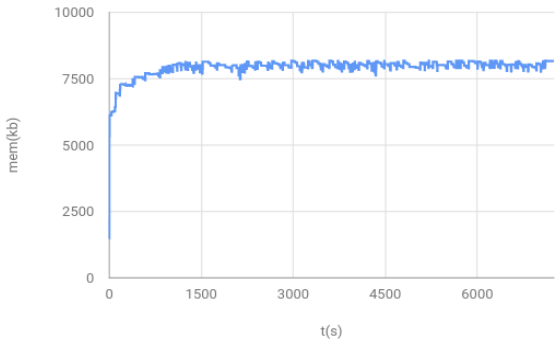


Figura 12. Uso de memória pela ferramenta de ataque na intensidade “Extreme”.

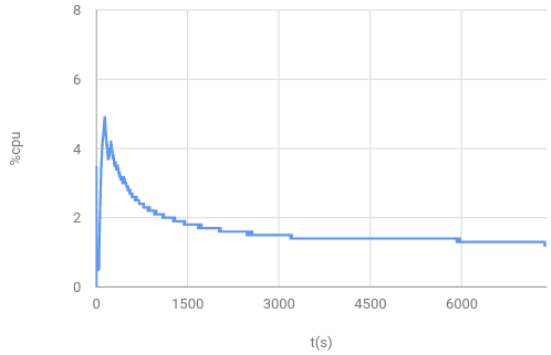


Figura 13. Uso de CPU pela ferramenta de ataque na intensidade “Extreme”.

Novamente, o uso de memória permaneceu em torno de 8 Mb, devido aos mesmos fatores anteriormente citados. A carga de CPU é infimamente maior devido ao aumento na quantidade de operações lógicas realizadas simultaneamente pelos 750 atacantes simulados; mesmo assim, a taxa de uso não superou os 5% (figuras 12 e 13).

6.2.4. Intensidade “Uber”

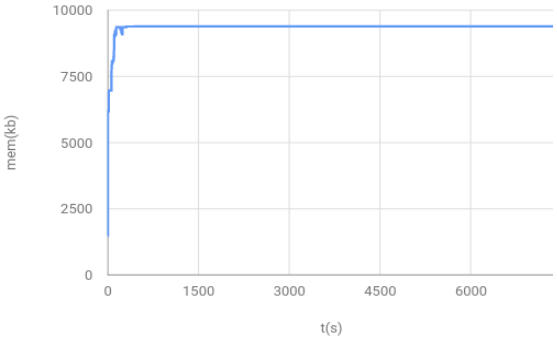


Figura 14. Uso de memória pela ferramenta de ataque na intensidade “Uber”.

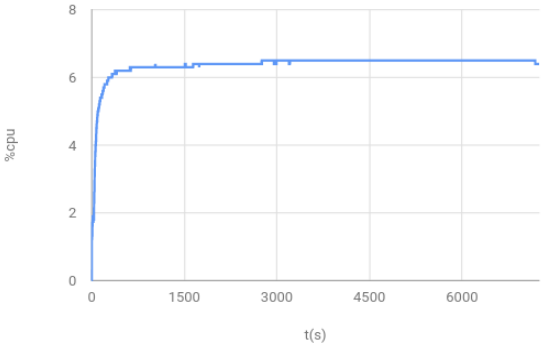


Figura 15. Uso de CPU pela ferramenta de ataque na intensidade “Uber”.

O uso de memória permaneceu próximo aos 10 Mb, maior que na intensidade “Extreme” devido à alocação de mais memória, necessária para as 1000 *threads* de ataque. O maior uso de CPU ocorre devido à sobrecarga causada pela virtualização de um número relativamente grande de *threads* (figuras 14 e 15).

6.3. Comportamento do Servidor Web Durante os Ataques

Foi medido o uso de memória e CPU do servidor web em operação normal e em caso de ataque que o indisponibiliza completamente. É possível perceber a eficiência do ataque devido ao grande aumento no uso de CPU e memória do Apache, causado pelos vários atacantes ocupando recursos do servidor.

6.3.1. Cenário Sem Ataque

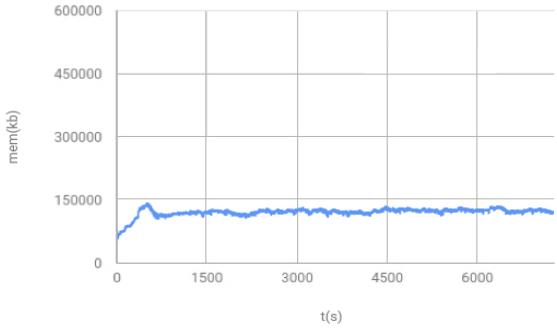


Figura 16. Uso de memória pelo servidor Apache em caso de operação normal.

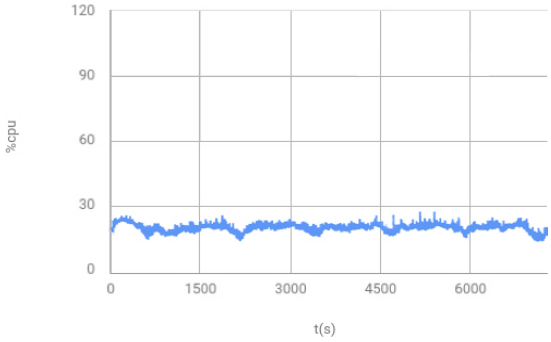


Figura 17. Uso de CPU pelo servidor Apache em caso de operação normal.

Percebe-se que, num cenário de utilização normal, com em torno de 33% do *buffer* de atendimento ocupado (NETCRAFT, 2016), o uso de memória por parte do servidor permanece praticamente constante e relativamente baixo, tendo em vista seu porte. O uso de CPU varia em um intervalo pequeno devido ao constante processo de gerência de *workers* para atenderem às requisições (figuras 16 e 17).

6.3.2. Cenário Com Ataque "Extreme"

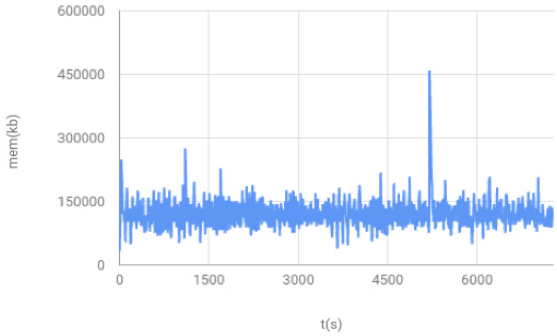


Figura 18. Uso de memória pelo servidor Apache em caso de indisponibilização total.

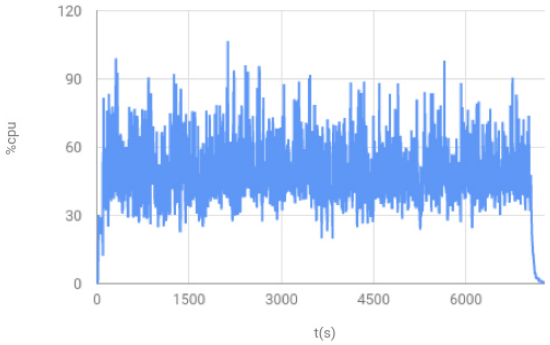


Figura 19. Uso de CPU pelo servidor Apache em caso de indisponibilização total.

O uso de memória pelo servidor em caso de indisponibilização chega a dobrar ou triplicar em pontos de pico, podendo prejudicar outros serviços que estão sendo executados na máquina hospedeira. Da mesma forma o uso de CPU, que chega a exorbitantes 95%, utilizando praticamente todo o poder computacional da máquina, demonstrando a efetividade da ferramenta móvel proposta (figuras 18 e 19).

7. CONSIDERAÇÕES FINAIS

A presente monografia teve como objetivo apresentar e avaliar uma implementação da versão móvel de uma das ferramentas de ataque *Slow* DoS mais populares na atualidade e na literatura, o *Slowloris*. Foi realizada uma descrição da solução proposta, descrevendo seus componentes básicos de arquitetura e sua forma de integração e uso.

Como fonte de conhecimento, foi adaptada a implementação *desktop* do *slowloris* para o domínio de dispositivos móveis, por meio de uma arquitetura de encapsulamento e passagem de parâmetros utilizando uma interface gráfica simples, baseada no projeto visual utilizado pelo *NeoLoris*. Por fim, foram realizados testes para avaliar o impacto do ataque no alvo e nos recursos do atacante.

Ficou demonstrado que, mesmo no ambiente limitado de um hospedeiro móvel e sem dispor de todas as funcionalidades de um *kernel* Linux padrão, a execução do *script* original por um dispositivo Android é viável em termos de desempenho e uso de recursos, podendo causar grandes prejuízos a serviços Web.

Com apenas uma instância em execução da ferramenta proposta, é possível indisponibilizar um serviço Web considerado de médio porte. Tal capacidade de realização e portabilidade de ataques permite que uma única pessoa possa utilizar dos seus dispositivos Android para ações maliciosas, prejudicando centenas de outros usuários que desejam acessar legitimamente os serviços disponíveis no alvo.

Além disso, fica evidente neste texto o poder de portabilidade das implementações atuais utilizando linguagens de *script*, denotando um cenário preocupante no que diz respeito ao surgimento e adaptação de ataques que tenham como vetores dispositivos móveis, caracterizando mais uma preocupação para os administradores de grandes serviços de rede.

A partir de tais resultados pretende-se, futuramente, validar o ataque utilizando outros tipos de redes móveis com diferentes latências como 2G, 3G e 4G LTE, além do aumento da escala das avaliações para números mais próximos de um ataque distribuído e/ou colaborativo real e o teste de outros ataques.

REFERÊNCIAS

A. KARIM; S. A. A. SHAH; R. B. SALLEH; M. ARIF; R. M. NOOR; S. SHAMSHIRBAND. Mobile Botnet Attacks – an Emerging Threat: Classification, Review and Open Issues. Ksii Transactions On Internet And Information Systems, V. 9, N. 4 , p. 1471–1488, 2015.

CAMBIASO, E.; PAPALEO, G; AIELLO, M. SlowDroid: Turning a Smartphone into a Mobile Attack Vector. FiCloud 2014, Barcelona,p. 405–408, 2014.

CAMBIASO, E.; PAPALEO, G.; CHIOLA, M.; AIELLO, M. Mobile executions of Slow DoS Attacks. Logic Jnl IGPL, V. 24, No. 1, p. 54–58, 16/10/2015.

CISCO. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015- 2020 White Paper. Relatório Técnico. Cisco, 2016. Disponível em: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html. Acesso em 20/11/2016.

COURSEN, S. The future of mobile malware. Network Security, EUA, v. 2007, n. 8, p. 7–11, 2007.

DANTAS, Y, G. Estratégias para tratamento de ataques de negação de serviço na camada de aplicação em redes IP. 2015. 77f. Trabalho de Conclusão de Curso (Especialização) Curso de Ciência da Computação, Universidade Federal da Paraíba, João Pessoa/PB, 2015.

DANTAS, Y, G.; NIGAM, V.; FONSECA, I. A Selective Defense for Application Layer DDoS Attacks, Intelligence and Security Informatics Conference (JISIC), Hague, 8/12/2014.

KARIM, A.; SALLEH, R.; SHAH, S. DeDroid: A Mobile Botnet Detection Approach Based on Static Analysis. IEEE, p. 1327–1332, 2015.

MICROSOFT. Microsoft Library: Security Issues with IP. Disponível em: https://technet.microsoft.com/en-us/library/cc959354.aspx. Acesso em 20/11/2016.

NETCRAFT. NetCraft WebServer Survey. Disponível em: https://news.netcraft.com/archives/2016/09/19/september-2016-web-server-survey.html. Acesso em 20/11/2016.

ZHAO, Y. Client/server two-way communication system framework under HTTP protocol. Patente de número US20020107910 A1, 08/10/2002.

PERL SCRIPT, Disponível em: https://pt.wikipedia.org/wiki/Perl. Acesso em 12/05/2017.

ANDROID, Disponível em: https://pt.wikipedia.org/wiki/Android. Acesso em 12/05/2017.

BRÄHLER, S. Analysis of the Android Architecture. Karlsuher Institut für Technologie, p. 1-25, 2010.

S. ALAM, “Apache released patch for ApacheKiller.pl Range Byte Flaw”. Disponível em: http://www.hackersgarage.com/apache-released-patch-for-apachekiller-pl-range-byte-flaw.html. Acesso em 12/05/2017

SLOWHTTPTEST, Disponível em: https://code.google.com/p/slowhttptest/. Acesso em 12/05/2017

WIFIKILL, Disponível em: http://forum.xda-developers.com/showthread.php?t=1282900. Acesso em 12/05/2017.

DROIDSHEEP, Disponível em: http://droidsheep.de/?page id=263. Acesso em 12/05/2017.

FACENIFF, Disponível em: http://faceniff.ponury.net. Acesso em 12/05/2017.

LOIC – Low Orbit Ion Cannon, Disponível em: https://play.google.com/store/apps/details?id=genius.mohammad.loic. Acesso em 12/05/2017.

PERLDROID. Disponível em: https://code.google.com/archive/p/perl-android-apk/. Acesso em 20/11/2016.

SIEGE. Disponível em: http://www. joedog. org/JoeDog/Siege. Acesso em 20/11/2016.

SLOWLORIS DDoS Attack. Disponível em: https://security.radware.com/ddos-knowledge-center/ddospedia/slowloris/. Acesso em 20/11/2016.

SYMANTEC. Symantec Internet Security Threat Report 2016, 2016. Disponível em: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf. Acesso em: 20/11/2016.

THE GUARDIAN. Major cyber attack disrupts internet service across Europe and US. Disponível em: https://www.theguardian.com/technology/2016/oct/21/ddos-attack-dyn-internet-denial-service. Acesso em 20/11/2016.

VISHRUT, S. An Analytical Survey of Recent Worm Attacks. International Journal of Computer Science and Network Security, EUA, p. 99-103, 2011.