



UNIVERSIDADE FEDERAL DO CEARÁ
Campus de Sobral

Curso de Engenharia da Computação

Francisco Wesley Melo Silva

**Utilização de dispositivos embarcados voltados para *IoT*
aplicados em ataques DDoS contra servidores Web**

Sobral – CE, 4 de novembro de 2018

FRANCISCO WESLEY MELO SILVA

Utilização de dispositivos embarcados voltados para *IoT* aplicados em ataques DDoS contra servidores Web

Trabalho de conclusão de curso apresentado ao Curso Engenharia de Computação como requisito parcial para obtenção do grau de bacharel em Engenharia de Computação na Universidade Federal do Ceará - *campus* Sobral.

Orientador

Prof. Me. Wendley Souza Silva

Sobral - CE, Novembro de 2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58u Silva, Francisco Wesley Melo.
Utilização de dispositivo embarcado voltado para IoT aplicados em ataques DDoS contra servidores Web / Francisco Wesley Melo Silva. – 2018.
40 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2018.
Orientação: Prof. Me. Wendley Souza Silva.

1. DDoS. 2. Servidor Web. 3. Internet. 4. Redes. 5. Internet das Coisas. I. Título.

CDD 621.39

FRANCISCO WESLEY MELO SILVA

Utilização de dispositivos embarcados voltados para *IoT* aplicados em ataques DDoS contra servidores Web

Trabalho de conclusão de curso apresentado ao
Curso Engenharia de Computação como requisito
parcial para obtenção do grau de bacharel em Enge-
nharia de Computação na Universidade Federal do
Ceará - *campus* Sobral.

Aprovado em: ____/____/_____.

BANCA EXAMINADORA

Prof. Me. Wendley Souza Silva (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. XXXXXXXXX XXXXXXX
Universidade Federal do Ceará (UFC)

Prof. Dr. XXXXXXXXX XXXXXXX
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

 Lorem ipsum mollis potenti quisque vitae aenean tempor purus, sociosqu vel nec aliquet eros id metus ut, purus integer ullamcorper pretium etiam curae enim. phasellus aliquam volutpat tristique sapien venenatis euismod taciti, sit ut maecenas ipsum curae sollicitudin, consequat curabitur elit accumsan ut quisque. pellentesque egestas dapibus feugiat laoreet a faucibus quisque proin lectus eleifend faucibus netus blandit, hac class ornare diam nibh condimentum sodales maecenas commodo fermentum sapien. ut nullam congue proin rutrum lectus vitae risus ac curabitur, suspendisse adipiscing consectetur tincidunt fermentum convallis eget ante feugiat vehicula, phasellus molestie luctus nullam convallis hac adipiscing vulputate.

 Et sapien pulvinar convallis aliquam potenti vehicula eget, luctus aliquam eu erat pretium ipsum dolor aenean, mattis ut volutpat posuere faucibus neque. aliquam nisl sem quisque ultricies turpis venenatis posuere dolor lorem, fermentum lorem venenatis curabitur fermentum nec orci phasellus, pretium nullam curabitur iaculis dui gravida taciti aenean. tempus lorem laoreet aliquam non pharetra posuere elit volutpat mollis, pellentesque sit class eu nec turpis felis facilisis eleifend, lobortis molestie augue convallis ac lacus gravida rhoncus. tristique arcu venenatis aliquam pharetra vulputate potenti eu pharetra, ac leo elementum commodo libero vel ac magna, lectus accumsan orci potenti tortor fringilla nullam condimentum, viverra mi massa nullam ad metus at.

 Hac lacus eu elementum feugiat conubia ipsum pharetra arcu, tincidunt aenean ultricies se-nectus aenean ut aliquam phasellus nam, eu ullamcorper nec arcu volutpat sapien ipsum. non potenti feugiat fermentum elit proin nam tempor, purus mattis dictum ante porta phasellus. ut habi-tasse vehicula adipiscing augue tempor tincidunt magna semper, nam aliquam id sociosqu cubilia dictumst erat volutpat, mattis potenti sed tortor tempor mollis turpis amet, cursus sit cras aenean iaculis dictum molestie. inceptos class consequat feugiat cras quisque felis blandit, sagittis tristique viverra curabitur imperdiet augue, cursus volutpat sodales nisi aliquam nam.

“DRACO DORMIENS NUNQUAM TITILLANDUS” - Lorem Ipsun

Lorem ipsum mollis potenti quisque vitae aenean tempor purus, sociosqu vel nec aliquet eros id metus ut, purus integer ullamcorper pretium etiam curae enim. phasellus aliquam volutpat tristique sapien venenatis euismod taciti, sit ut maecenas ipsum curae sollicitudin, consequat curabitur elit accumsan ut quisque. pellentesque egestas dapibus feugiat laoreet a faucibus quisque proin lectus eleifend faucibus netus blandit, hac class ornare diam nibh condimentum sodales maecenas commodo fermentum sapien. ut nullam congue proin rutrum lectus vitae risus ac curabitur, suspendisse adipiscing consectetur tincidunt fermentum convallis eget ante feugiat vehicula, phasellus molestie luctus nullam convallis hac adipiscing vulputate. Et sapien pulvinar convallis aliquam potenti vehicula eget, luctus aliquam eu erat pretium ipsum dolor aenean, mattis ut volutpat posuere faucibus neque. aliquam nisl sem quisque ultrices turpis venenatis posuere dolor lorem, fermentum lorem venenatis curabitur fermentum nec orci phasellus, pretium nullam curabitur iaculis duis gravida taciti aenean. tempus lorem laoreet aliquam non pharetra posuere elit volutpat mollis, pellentesque sit class eu nec turpis felis facilisis eleifend, lobortis molestie augue convallis ac lacus gravida rhoncus. tristique arcu venenatis aliquam pharetra vulputate potenti eu pharetra, ac leo elementum commodo libero vel ac magna, lectus accumsan orci potenti tortor fringilla nullam condimentum, viverra mi massa nullam ad metus at.

Palavras-chave: lorem, ipsum, dolor.

ABSTRACT

Lorem ipsum mollis potenti quisque vitae aenean tempor purus, sociosqu vel nec aliquet eros id metus ut, purus integer ullamcorper pretium etiam curae enim. phasellus aliquam volutpat tristique sapien venenatis euismod taciti, sit ut maecenas ipsum curae sollicitudin, consequat curabitur elit accumsan ut quisque. pellentesque egestas dapibus feugiat laoreet a faucibus quisque proin lectus eleifend faucibus netus blandit, hac class ornare diam nibh condimentum sodales maecenas commodo fermentum sapien. ut nullam congue proin rutrum lectus vitae risus ac curabitur, suspendisse adipiscing consectetur tincidunt fermentum convallis eget ante feugiat vehicula, phasellus molestie luctus nullam convallis hac adipiscing vulputate. Et sapien pulvinar convallis aliquam potenti vehicula eget, luctus aliquam eu erat pretium ipsum dolor aenean, mattis ut volutpat posuere faucibus neque. aliquam nisl sem quisque ultrices turpis venenatis posuere dolor lorem, fermentum lorem venenatis curabitur fermentum nec orci phasellus, pretium nullam curabitur iaculis duis gravida taciti aenean. tempus lorem laoreet aliquam non pharetra posuere elit volutpat mollis, pellentesque sit class eu nec turpis felis facilisis eleifend, lobortis molestie augue convallis ac lacus gravida rhoncus. tristique arcu venenatis aliquam pharetra vulputate potenti eu pharetra, ac leo elementum commodo libero vel ac magna, lectus accumsan orci potenti tortor fringilla nullam condimentum, viverra mi massa nullam ad metus at.

Sumário

1	Introdução	5
2	Objetivos	8
2.0.1	Objetivos Gerais	8
2.0.2	Objetivos Específicos	8
3	Fundamentação teórica	9
3.1	Arquitetura cliente-servidor	9
3.1.1	Web Server	9
3.2	<i>Internet of Things</i>	9
3.2.1	Dispositivos embarcados	11
3.2.2	ESP8266/NodeMCU	12
3.2.3	ESP32	13
3.3	Ataque de negação de serviço	14
3.3.1	Ataque DoS	14
3.3.2	ICMP <i>echo request</i>	15
3.3.3	DDoS	16
3.4	<i>Sniffer</i>	17
3.4.1	Wireshark	18
4	Metodologia	19
5	Experimentação	20

5.1	Algoritmo	20
5.2	Execução do experimento	20
6	Resultados obtidos	24
7	Conclusão	27
	Referências	28

Lista de Figuras

1	Estrutura da Internet de forma simplificada	6
2	Exemplo de ataque de negação de serviço	7
3	NodeMCU	13
4	ESP32	14
5	Ataque de <i>flood</i> ICMP	16
6	Fluxo de tráfego durante ataque DDoS	17
7	Diagrama do funcionamento de <i>sniffer</i>	18
8	Interface gráfica do Wireshark	19
9	Lista de Clientes conectados	22
10	Página Web entregue pelo servidor Web	22
11	Dispositivos utilizados para o ataque DDoS	23
12	Fluxo de pacotes com destino ao servidor web	23
13	Fluxo de bytes com destino ao alvo	24
14	Tela do navegados contendo erro	24
15	Chaveamento de processos no sistema operacional.	26

Lista de Tabelas

1	Características de hardware ESP8266	12
2	Características do ESP32	13
3	<i>Hardware</i> dedicado para a máquina virtual	19

Lista de Algoritmos

1	Envio constante de requisicoes ICMP para um host na rede	21
2	Algoritmo utilizado no ataque	30

Lista de Siglas

ACK	: <i>Acknoledge Message</i> - Mensagem de Reconhecimento
ADC	: <i>Analog Digital Converter</i> - Conversor Analógico-Digital
CPU	: Central Processing Unit - Unidade Central de Processamento
DAC	: <i>Digital Analog Converter</i> - Conversor Digital-Analógico
DDoS	: <i>Distributed Denial of Service</i> - Negação Distribuída de Serviço
DoS	: <i>Denial of Service</i> - Negação de Serviço
GPIO	: <i>General Purpose Input Output</i> - Propósito Geral de Entrada e Saída
HTML	: <i>HyperText Markup Language</i> - Linguagem de Marcação de Hipertexto
HTTP	: <i>HyperText Transfer Protocol</i> - Protocolo de Transferência de Hipertexto
ICMP	: <i>Internet Control Message Protocol</i> - Protocolo de Mensagem de Controle da Internet
IDE	: <i>Integrated Development Environment</i> - Ambiente de Desenvolvimento Integrado
IoT	: <i>Internet of Things</i> - Internet das Coisas
NIC	: <i>Network Interface Card</i> - Placa da Interface de Rede
PWM	: <i>Pulse Width Modulation</i> - Modulação de Tamanho de Pulso
RAM	: <i>Random Access Memory</i> - Memória de Acesso Randômico
RFC	: <i>Request for comment</i> - Requisita comentários
SRAM	: <i>Static Random Access Memory</i> - Memória de Acesso Randômico Estático
SYN	: <i>Synchronize Message</i> - Mensagem de Sincronização
TCP	: <i>Transmission Control Protocol</i> - Protocolo de Controle de Transmissão
UDP	: <i>User Datagram Protocol</i> - Protocolo de Datagrama do Usuário
USB	: <i>Universal Serial Bus</i> - Barramento Serial Universal

1 Introdução

Com a evolução dos sistemas de comunicação, muitas aplicações foram desenvolvidas e incorporadas ao cotidiano. Sistemas capazes de prover o funcionamento de diversos serviços na *web* como atendimento bancário, *streaming* de mídia, serviços de localização, agenciamento de transporte, etc. Ao passo desse desenvolvimento outra onda tecnológica tomou força nos últimos anos, a chamada IoT (*Internet of Things* ou Internet das Coisas) tem sido capaz de suportar cenários cada vez mais interativos, integrando diferentes redes e dispositivos, tudo isso baseado na *web*.

Essa grande rede está distribuída ao redor do globo, composta por diversos dispositivos de variado porte que vão desde grandes *mainframes*, a simples dispositivos embarcados. Isso torna esse sistema altamente escalável e tolerante a falhas porém, também possibilita que seus nós sejam utilizados para outros fins, às vezes com propósitos destrutíveis.

Com potencial para causar graves problemas em diversos cenários, como no mercado financeiro, controle de rodovias, processos industriais, e especialmente em sistemas de comunicação em tempo real do tipo *hard real-time* que, segundo Kopetz (1997, p.3) são sistemas os quais em uma única falha pode ter consequências catastróficas, por exemplo no controle de tráfego aéreo, ou sistemas de proteção contra desastres naturais.

Primordialmente, é necessário definirmos as tecnologias que podem ser utilizadas para o funcionamento da Internet, assim como dos dispositivos que compõe IoT e suas características principais, posteriormente serão descritas as ferramentas e técnicas utilizadas para obtenção dos dados que serão analisados.

O modo como as redes que compõe a Internet foi estruturado há muito tempo, de acordo com Pastor-Sorras *et al* (2007, p.3) trata-se de um sistema distribuído de redes menores que podem transmitir informações entre si. As informações transferidas entre computadores são divididas em pacotes que são repassados para dispositivos que tem a função de transferi-los entre diferentes redes, são os chamados *roteadores*.

Ainda em Pastor-Satorras *et al* (2007, p.4), um roteador não está diretamente ligado a todos os outros roteadores que compõe a Internet. Ele apenas tem a função de definir que “direção” os dados irão percorrer para chegar ao destinatário, além de reorganizar o tráfego de informações quando

um dos nós da rede – computadores, roteadores, servidores, etc. – não estão funcionando devidamente. A figura 1 mostra de forma simplificada o funcionamento da Internet e sua intersecção de redes.

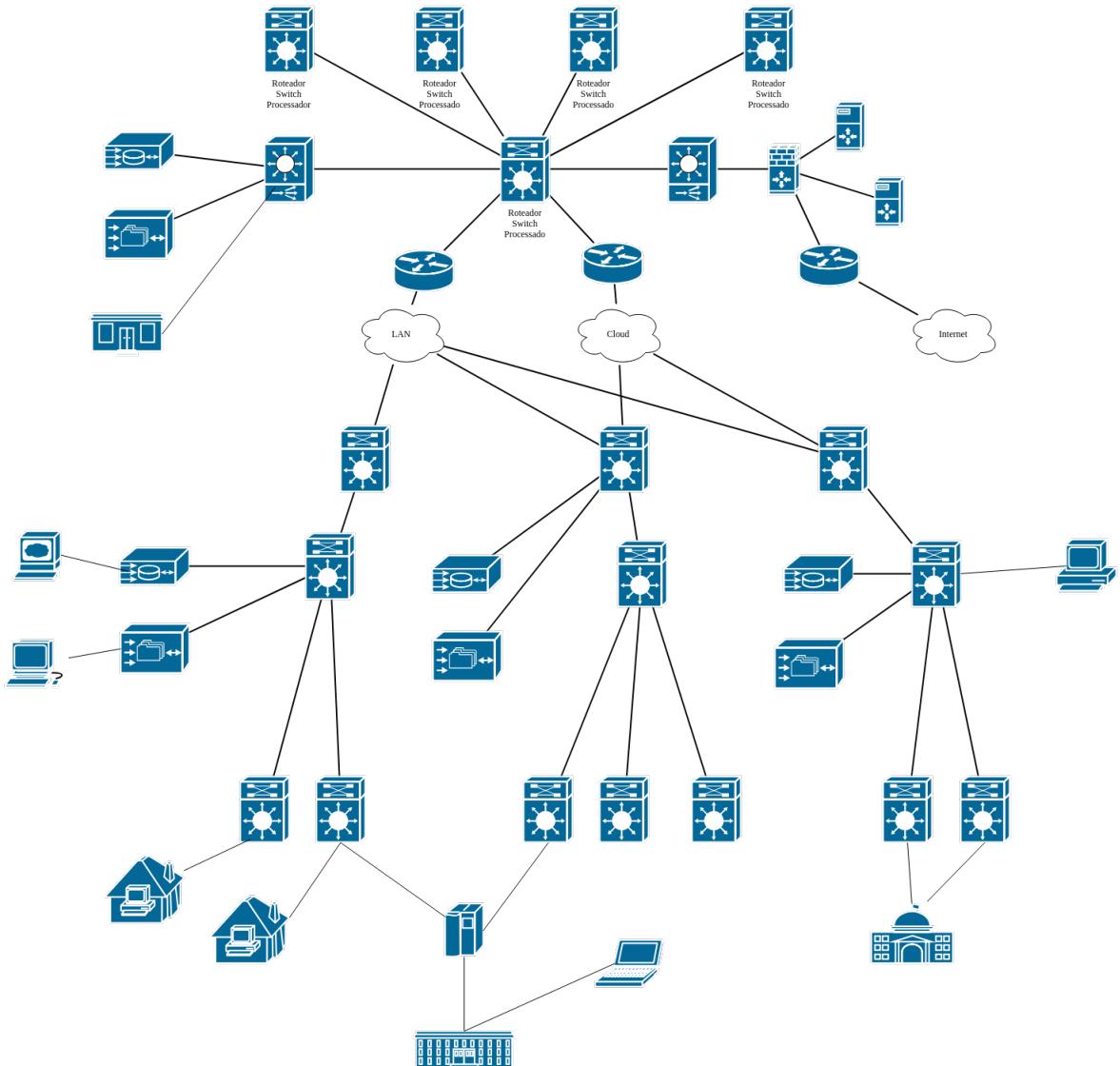


Figura 1: Estrutura da Internet de forma simplificada

fonte: o autor.

A grande quantidade de rotas que existe para enviar uma mensagem a partir de um nó para outro possibilita que uma delas possa ser utilizada com propósito de desestabilizar nós específicos como servidores, ou mesmo roteadores tornando incomunicável um vasto conjunto de nós. Isso acontece devido às falhas de segurança, que permitam que um grande fluxo de requisições seja direcionado para um único destinatário.

Contudo, apenas uma falha de segurança não é suficiente para que seja performada uma ofensiva. Um ataque de negação de serviço, por exemplo, utiliza diversos dispositivos conectados numa rede para direcionar um fluxo de requisições. Diversos tipos de dispositivos podem ser utilizados; desde computadores convencionais, presentes nas casas das pessoas, até mesmo objetos inteligentes conectados à Internet, como demonstrará este trabalho.

A figura 2, exemplifica um conjunto de nós de uma rede, executando um ataque a um servidor Web convencional.

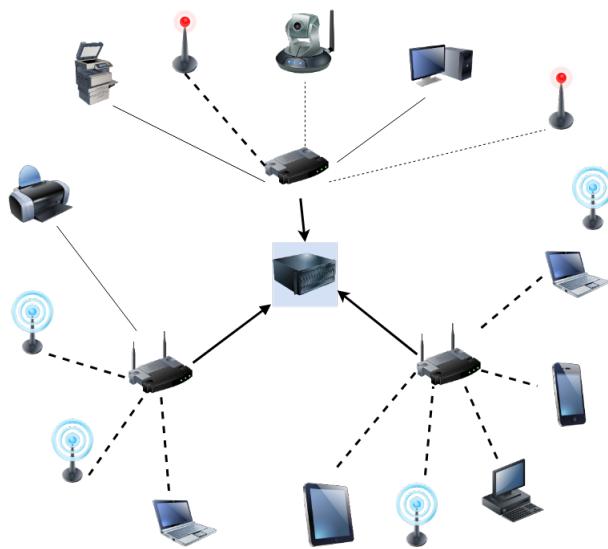


Figura 2: Exemplo de ataque de negação de serviço

fonte: O autor.

2 Objetivos

2.0.1 Objetivos Gerais

Este trabalho tem como objetivo principal fazer uma análise de um cenário em que dispositivos empregados para a construção de redes IoT, podem ser utilizados de forma destrutiva para atacar, desestabilizar e até mesmo desabilitar serviços de Internet que são de grande importância para o mundo atual.

Analisar um cenário em que tecnologias usadas para IoT são aplicadas de forma que sejam utilizadas para ataque a sistemas Web, e tentativa de desativação dos sistemas através de ataque de negação de serviço (DoS).

2.0.2 Objetivos Específicos

- Utilizar dados coletados durante o ataque para a extração de dados a respeito da robustez dos servidores web;
- Propiciar uma análise crítica a respeito das brechas de seguranças em dispositivos IoT.

3 Fundamentação teórica

3.1 Arquitetura cliente-servidor

3.1.1 Web Server

Segundo Morimoto (2015, p.347), os servidores *web* são a espinha dorsal da Internet, são eles que hospedam todas as páginas, incluindo os mecanismos de busca e servem como base para todo aplicativo via web. Um dos principais softwares que executam essa função é o Apache, que suporta desde os recursos básicos como prover uma simples página em HTML, à expansão do suporte através de módulos para linguagens de programação, bancos de dados, etc. Existem vários tipos de servidores voltados para web e diversas aplicações, como e-mail, armazenamento e transferência de arquivos, armazenamento de banco de dados, etc.

Neste trabalho foi utilizado o Apache da família 2.x que se destaca pelo seu desempenho e flexibilidade.

Segundo Tanenbaum *et al* (2006, p.556), o servidor web Apache é um software completo com diversas melhorias para suportar os tipos de arquivos presentes na web, com configuração profundamente personalizáveis e, ao mesmo tempo independente de plataformas específicas, ou seja, capaz de prover seu próprio ambiente de desenvolvimento e posteriormente implementá-lo em outros sistemas operacionais.

Ainda em Tanenbaum *et al* (2006, p.557), é informado que devido os navegadores web, os servidores *web* utilizam HTTP como protocolo de comunicação. O HTTP é virtualmente implementado no topo da pilha TCP, e por essa razão o núcleo do Apache assume que todos as requisições são do tipo TCP; e por padrão, qualquer requisições UDP não é processada sem que antes haja modificações no núcleo do Apache.

3.2 Internet of Things

Em Buyya (2016, p.10), *Internet of Things* é definido por dois pilares importantes: “Internet” e “Coisas”. O autor ressalta ainda que nem todos os objetos capazes de se conectar à Internet entrarão na categoria de “coisa” pois, essa notação é usada para encapsular um conjunto genérico

de entidades, incluindo dispositivos inteligentes, sensores e até mesmo seres humanos; tornando-se acessível à qualquer momento e a partir de qualquer lugar. Isso implica que objetos devem se conectar sem qualquer restrição.

Segundo Buyya (2016, p.11), é crucial a conectividade ubíqua para IoT, e para atender as aplicações necessárias para suportar um conjunto diverso de dispositivos e protocolos de comunicação, oriundos de sensores capazes de detectar e reportar a mensagem necessária para poderosos servidores, utilizados para armazenamento, processamento e extração de conhecimentos à partir dessas informações. Isso requer um integração plena dos dispositivos móveis, apetrechos responsáveis pela comunicação, como roteadores, e claro os seres humanos como controladores.

Ainda segundo o autor, a tecnologia dominante no início das aplicações comerciais utilizando IoT foi a identificação por Rádio-Frequência (RFID) porém, com o avanço das tecnologias de comunicação, das redes de sensores sem fio, e dispositivos dotados de *Bluetooth*, ocasionaram uma mudança nesse cenário. Além de tornar viável o atendimento aos requisitos essenciais de sistemas de comunicação em tempo real, tais como: escalabilidade, suporte à heterogeneidade, integração total, e processamento em tempo real.

De acordo com daCosta *et al* (2013, p.23), o desenvolvimento dessa nova arquitetura é proveniente da Internet tradicional e outras tecnologias que propõem princípios básicos, tais como:

- Ser o mínimo específico, deixando a arquitetura mais aberta possível;
- Os sistemas devem ser projetados para falhar da forma menos impactante possível;
- Os graus de complexidade e funcionalidade da rede são aplicados somente quando necessários;
- A arquitetura é criada a partir de conceitos simples que são construídos em sistemas complexos usando sinais analógicos providos de fenômenos naturais;
- Possibilitar que dados sejam extraídos em tempo real.

A arquitetura emergente da Internet das Coisas pretende ser mais inclusiva para um mercado de maior variedade, reduzindo as informações que trafegam na rede. Ela deve ser extremamente tolerante às falhas, erros e conexões intermitentes naquele nível.

3.2.1 Dispositivos embarcados

Um sistema embarcado é dispositivo eletrônico desenvolvido com um propósito muito específico. Um sistema embarcado consiste em um microprocessador que é interconectados com outros componentes discretos. Cada componente é selecionado baseado nos requisitos funcionais do sistema.

Frequentemente, sistemas embarcados menores são usados para controlar um comportamento de aspecto específico de um sistema muito mais complexo. Sistemas embarcados executam um papel diferente de um computador de propósito geral, como os computadores de mesa (*desktops*). O usuário final não é capaz alterar como o dispositivo funciona.

Suas principais vantagens são:

- Baixo custo;
- Desempenho aprimorado;
- Confiabilidade.

Segundo Shukla (2016, p.15), há quatro categorias de sistemas embarcados, são eles:

- Computação geral, com aplicação semelhante aos computadores de mesa, mas em *hardware* embarcado;
- Sistemas de controle, aplicado em sistemas de controle com resposta em tempo real;
- Processamento de sinais, aplicado na computação envolvendo grandes taxas de transmissão;
- Comunicação e Redes, empregado na infraestrutura de comunicação, em roteadores, etc.

Neste trabalho foi utilizado a plataforma de prototipagem NodeMCU, que consiste em microprocessador embarcado capaz de se conectar facilmente às redes de comunicação.

3.2.2 *ESP8266/NodeMCU*

De acordo com Kodali *et al* (2017, p.4), trata-se de uma solução altamente integrada e de baixo consumo. Composto por GPIOs, um chip 12C, ADC, DAC, PWM, Wi-Fi, regulador de tensão e uma conexão USB).

A tabela 1 expõe algumas características do ESP8266:

Tabela 1: Características de hardware *ESP8266*

Componente	Detalhes
Processador	Tensilica L-106 32 bits 80 MHz
Voltagem operacional	2.5V ~3.6V
Corrente operacional	~ 80mA
Frequência de operação (Wi-Fi)	2.4 GHz ~ 2.5 GHz Tx/Rx
Protocolos de rede	IPv4, TCP/UDP/HTTP/FTP
Segurança	WPA2
Encriptação	WEP/TKIP/AES
Memória RAM	64 KB de memória para instruções
Memória ROM	96 KB de memória para dados
GPIO	16 pinos

fonte: Manual fornecido pelo fabricante.

Com aparatos físicos Sua aplicação é semelhante à da plataforma de prototipagem Arduino, inclusive é possível utilizar a IDE do Arduino para codificar instruções para o ESP8266, embora oficialmente seja indicado que se utilize a linguagem *Lua*. A figura 3, ilustra o dispositivo NodeMCU utilizado no experimento.

Neste trabalho foram utilizados 8 (oito) unidades do ESP8266.



Figura 3: NodeMCU

fonte: Espressif (2018)

3.2.3 ESP32

Em Allafi (2017, p.2), o EPS32 é descrito como um microcontrolador *open-source* baseado numa placa de *Input* e *Output*. A placa é desenvolvida para suportar Wi-Fi, e conta com 36 GPIO.

Sua aplicação é semelhante às do ESP8266, porém, com um hardware mais robusto.

A tabela a seguir mostra algumas características básicas do ESP32.

Tabela 2: Características do ESP32

Item	Descrição
Voltagem operacional	2.2V ~ 3.6V
GPIO	36 pinos
ADC	14 pinos
DAC	2 pinos
Memória	16 MB
SRAM	250 KB
Processador	Xtensa dual-core 32 bits 160 MHz
Frequência de operação (Wi-Fi)	2.4 GHz
Conectividade	Wi-Fi e Bluetooth

fonte: Espressif (2018)

Neste trabalho foi utilizado 2 (duas) unidades do ESP32, da marca WROOM. A figura 4,

ilustra o dispositivo ESP32 utilizado no experimento.



Figura 4: *ESP32*

fonte: Espressif (2018)

3.3 Ataque de negação de serviço

3.3.1 Ataque DoS

Existem diversas técnicas de ataques DoS. Neste trabalho foi utilizada a técnica de *Flooding* TCP SYN. De acordo com Jian (2000, ? apud Gu 2007, p.5) esses ataques frequentemente exploram protocolos de rede estruturados, devido esses protocolos consumirem recursos para manter metadados. O *flooding* TCP SYN é um dos tipos de ataque que tem grave impacto em muitos sistemas.

Quando um cliente tenta estabelecer uma conexão TCP com o servidor, o cliente inicialmente envia uma mensagem SYN para o servidor. De acordo com Filho (2013, p.141) as mensagens SYN são usadas para iniciar uma conexão e as mensagens ACK servem para que um dos lados, ao receber uma mensagem sobre um pacote, confirme, para quem o enviou, que sabe o número do pacote seguinte e confirmar que recebeu o anterior. Jian (2000, ? *apud* Gu 2007, p.5) relata que o servidor então informa o conhecimento dessa mensagem ao enviar uma mensagem SYN-ACK.

O cliente completa o estabelecimento da conexão ao responder com outra mensagem ACK. A conexão entre o cliente e o servidor está então aberta, e os dados podem ser trafegados entre eles. O abuso dessa conexão acontece quando a conexão está “semiaberta”. O servidor precisa alocar memória para armazenar as informações de cada conexão, e a memória não será desocupada até que o servidor receba a mensagem ACK final ou a conexão “semiaberta” expire.

Os *hosts* atacantes são capazes de criar uma grande quantidade de conexões através da falsificação de IPs em mensagens SYN, ou ignorando as SYN-ACKs, repetindo o processo de abertura de conexão.

O ICMP *Smurf Flooding* é frequentemente usado para determinar se um computador numa rede está respondendo. Para que isso ocorra, um pacote de requisição ICMP *echo* é enviado ao computador. Na seção seguinte, esse processo será explicado com mais detalhes.

3.3.2 ICMP echo request

De acordo com Filho (2013, p.212) os ataques de *flood* por *ping* (ICMP *echo request*) são realizados quando os nós de uma rede realizam grande quantidade de requisições do tipo *ping*, com o intuito de causar estresse em computadores servidores. O cabeçalho desse datagrama possui 32 bits, sendo divididos da seguinte forma, os primeiros 8 bits determinam o seu tipo, e os 8 bits posteriores o seu código, e os 16 bits restante estão relacionados ao *checksum*. Neste trabalho foi utilizado a requisição de tipo 8 - *echo request* e código '0 - ping echo request'.

Ao enviar múltiplas requisições por segundo, a máquina servidora tentará interpretá-las e respondê-las, porém, devido a suas limitações físicas nem todas as solicitações poderão ser respondidas, podem ser constatadas instabilidades, esse cenário se agrava quando múltiplos *hosts* estão fazendo grande quantidade de requisições para essa máquina.

A figura 5, a seguir ilustra como esse tipo de ataque acontece.

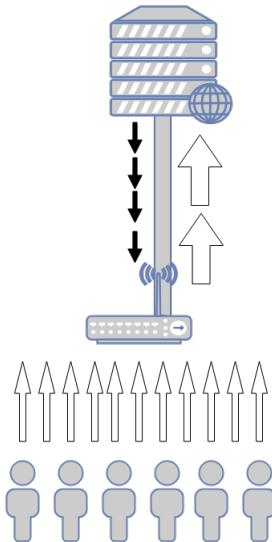


Figura 5: Ataque de flood ICMP

fonte: O autor.

Outra forma de ataque utilizando ICMP acontece de acordo com CERT (1998 ? *apud* Gu 2007, p.212), quando um *host* é capaz de forjar requisições ICMP tendo o endereço da vítima como fonte do ataque e o endereço de *broadcast* dessas redes como o endereço de destino. Se o *firewall* ou roteador da rede não for capaz de filtrar todos os pacotes, eles serão entregues a todas as máquinas daquela rede. Essas máquinas enviarão respostas ao ICMP *echo* de volta para a fonte (que foi adulterada e é vítima). Assim, a rede e a máquina atacada estarão congestionadas.

3.3.3 DDoS

Em Gu *et al* (2007, p.8) é relatado que vários ataques são feitos a partir de diversos *hosts* num sistema distribuído. Geralmente um ataque DDoS ocorre em duas fases. Primeiro, um atacante constrói uma rede de ataque que é constituída por milhares de dispositivos, conhecidos também como “zumbis” ou “bots”. Então o atacante causa um tremendo volume de tráfego contra a vítima.

Para construir uma rede de ataque, o atacante procura por dispositivos que estão desprotegidos. Em geral, um *host* pode ser comprometido de duas formas.

Sendo a primeira em que o usuário é enganado e forçado a executar programas maliciosos, tais como: vírus, *spywares* etc. Ao executar esses programas, eles podem fazer a máquina atuar

como “bot”, fazendo parte de uma *botnet*. Quando houver o comando por parte do mestre (aquele que comanda a rede), todos os “bots” enviarão requisições ICMP para o alvo.

A segunda forma é também à partir de programas maliciosos que escaneiam e exploram vulnerabilidades em computadores remotos. Tais vulnerabilidades permitem que o atacante adentre à máquina e instale *softwares* para que a máquina infectada além de fazer parte da *botnet*, passe também a procurar por vulnerabilidades em dispositivos para fazer deles, parte da rede atacante.

A figura 6 a seguir ilustra o tráfego durante um ataque DDoS.

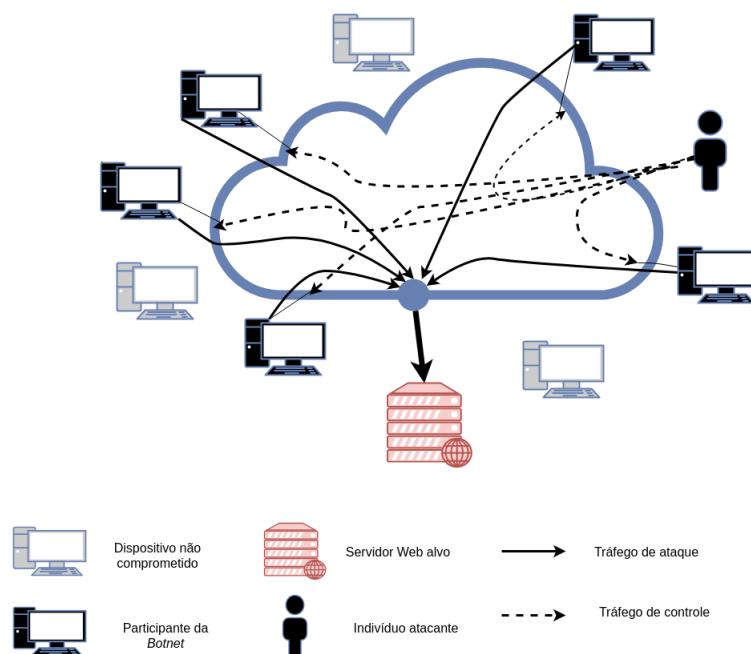


Figura 6: Fluxo de tráfego durante ataque DDoS

fonte: O autor.

3.4 Sniffer

Segundo Asrodia *et al* (2013, p.1), *sniffing* de pacotes é o processo de capturar as informações transmitidas através de uma rede. Nesse processo, o NIC captura todo o tráfego que flui dentro ou fora de uma rede. Essa prática é principalmente usada no gerenciamento de redes, monitoramento e o *ethical hacking*. Para executar o *sniffing* são utilizados os chamados *sniffers* de pacotes.

Ainda segundo o autor, quando um pacote é transferido de um nó remetente a um nó destinatário, geralmente passa por muitos dispositivos intermediários. Um nó cujo NIC está configurado em modo 'promíscuo' recebe todas as informações que percorre a rede. Cada NIC tem um endereço físico que é diferente de outro, e de outra rede.

Quando um pacote chega ao NIC, o endereço do *hardware* do *frame* é combinado com o endereço físico que o NIC tem, mas se o definirmos no modo 'promíscuo', todos os pacotes chegarão naquela interface. Quando é utilizado um *switch* que já passam os dados filtrados então é utilizado uma técnica para capturar todos os dados da rede. Quando o NIC aceita os pacotes, eles são copiados para a memória e posteriormente para o *kernel* e em seguida para a aplicação utilizada pelo administrador da rede.

A figura 7 a seguir mostra o esquema do funcionamento de um *sniffer*.

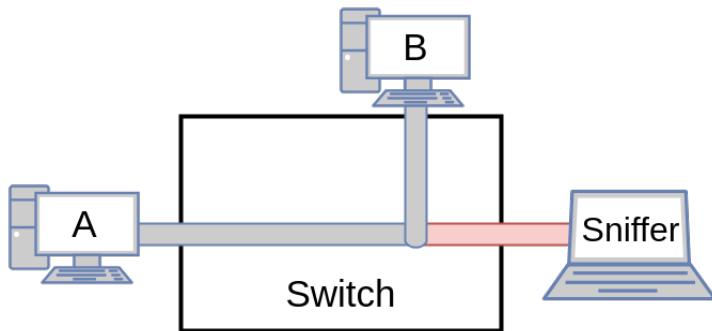


Figura 7: Diagrama do funcionamento de sniffer

fonte: O autor.

Existem diversos *softwares* que executam essa função, tais como Wireshark, TCPDump e Nmap. Neste trabalho foi utilizado o *sniffer* Wireshark.

3.4.1 Wireshark

De acordo com Goyal *et al* (2017, p.1), o Wireshark é a ferramenta de *sniffing* mais popular. Trata-se de uma interface gráfica simples dotada de opções de filtragem dos pacotes capturados. É uma ferramenta gratuita, assim como a maioria dos demais *sniffers* disponíveis. Ele tem a capacidade de *scanear* qualquer tipo de redes Ethernet, Wi-Fi e até mesmo Bluetooth. Essa ferramenta pode rapidamente detectar e reportar ataques DoS, em firewalls ou qualquer brecha de segurança em redes.

Além disso, é capaz de percorrer profundamente cada pacote e separar diferentes pacotes por protocolo de rede.

A seguinte figura (8) mostra a interface gráfica do Wireshark.

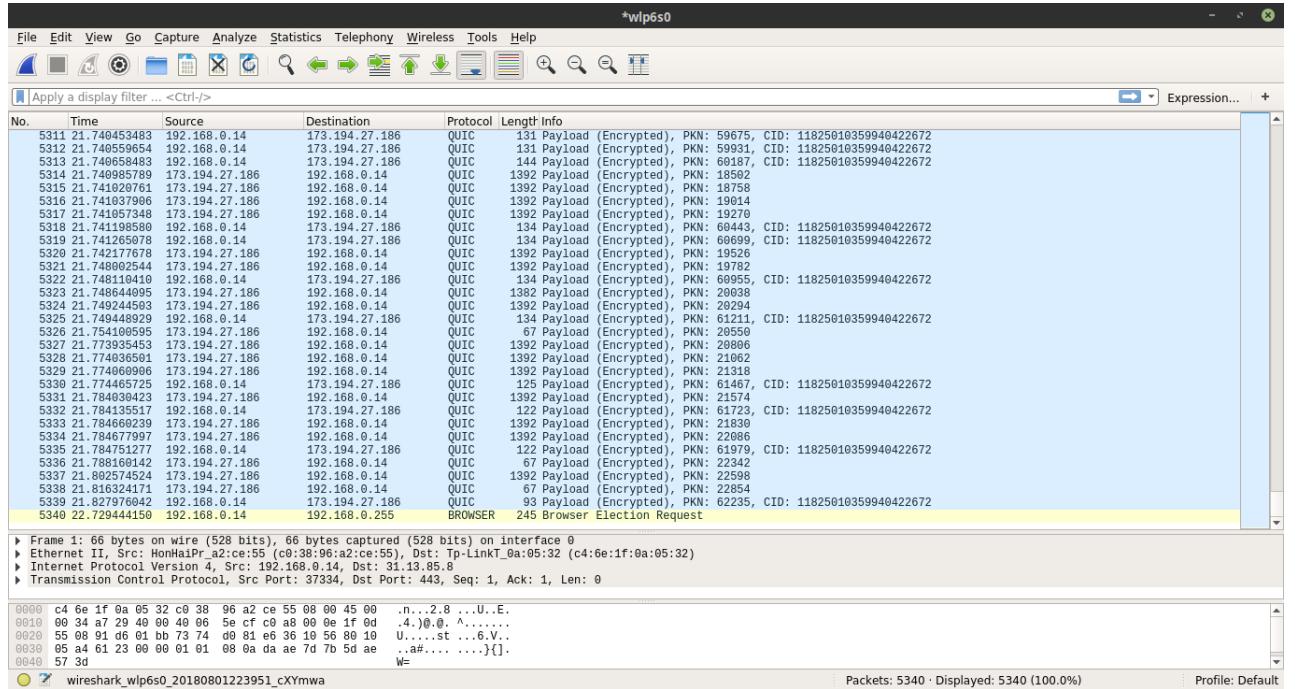


Figura 8: Interface gráfica do Wireshark

fonte: O autor.

4 Metodologia

Foi utilizado uma máquina dedicada que executava o sistema operacional *Ubuntu Server* na versão 16.04.5 LTS e o servidor web Apache 2. O *hardware* disponível em tal máquina está descrito na tabela 3, a seguir.

Tabela 3: Hardware dedicado para a máquina virtual

Hardware:	Descrição:
Processador	Intel Pentium Dual Core
Memória RAM	2 GB
Armazenamento	40 GB
Placa de rede	Placa de rede <i>on-board</i> Ethernet

fonte: O autor.

Como citado anteriormente, foram utilizadas 8 unidades do NodeMCU e 2 unidades do ESP32, que atuaram como computadores que faziam requisições ICMP constantemente.

Após a gravação do código no microprocessador embarcado no NodeMCU e no ESP32, são definidas alguns detalhes. Inicialmente, na função *setup()* são colocadas instruções que tratarão da conexão do dispositivo a uma rede Wi-Fi. Ainda sobre o código, na função *loop()*, estão guardadas as rotinas que de fato farão as requisições ao servidor *Web*. Uma característica do NodeMCU é que semelhante à plataforma *Arduino* as instruções da função *loop()*, são executadas repetidamente do começo ao fim, a menos que ocorra um evento que force a interrupção, por exemplo: falta de energia, ou erro não tratado pelo código.

Num computador conectado à mesma rede, o *software Wireshark* é executado, atuando como *Sniffer*, ele tem a função de analisar os pacotes que trafegam na rede. Neste trabalho, o *Wireshark* foi definido para procurar por pacotes ICMP, tanto de requisição como respostas (*echo requests* e *echo responses*). O mesmo conta com ferramentas próprias que fazem análises estatísticas a respeito do tráfego. E dados obtidos podem ser exportados para outros *softwares* com funções matemáticas específicas, como o *SciLab*.

Após serem ligados os NodeMCUs e o ESP32, foi iniciado um cronômetro externo, de modo a contabilizar o intervalo de tempo entre eventuais acontecimentos ocorridos durante o experimento, tais como instabilidade da máquina servidora, parada total de funcionamento da mesma, etc.

5 Experimentação

5.1 Algoritmo

A seguir está descrito em pseudo-código, o algoritmo utilizado para executar requisições ICMP.

5.2 Execução do experimento

A figura 9, a seguir mostra os dispositivos conectados à rede em que está o servidor vítima do ataque DDoS.

Algorithm 1 Envio constante de requisicoes ICMP para um host na rede

```
ssid ←' Id – da – rede'  
password ←' Senha'  
  
void():  
    ComunicacaoSerial.inicia("115200 baulds")  
    Aguardar(10-milissegundos)  
  
    imprimir(Conectando a rede...)  
    Wifi.inicia(ssid, password)  
    return null  
  
while Wifi status != conectado do  
    esperar(100-milissegundos)  
end while  
  
MonitorSerial.imprimir('Wifi-conectado-com-ip')  
MonitorSerial.imprimir(Wifi.IpLocal())  
  
loop():  
    MonitorSerial.imprimir('ping-host')  
    MonitorSerial.imprimir(endereço-host)  
  
if resposta recebida == verdadeiro then  
    MonitorSerial.imprimir('Sucesso')  
    ELSE  
    MonitorSerial.imprimir('Erro!')  
end if  
    esperar(500-milissegundos)  
return null
```

Figura 9: Lista de Clientes conectados

Listagem de Clientes DHCP				
ID	Nome do Cliente	Endereço MAC	Atribuído IP	Tempo de Renovação
1	android-2676637939	00-0A-F5-42-0D-20	192.168.0.2	01:49:32
2	Unknown	1C-5A-6B-2E-B8-35	192.168.0.14	Permanente
3	Unknown	00-55-4E-9E-2A-89	192.168.0.15	Permanente
4	android-75ec35d70d05b6db	A8-96-75-44-6A-75	192.168.0.4	00:11:44
5	ESP_PL5F0R	80-58-F8-3C-C4-2A	192.168.0.5	00:18:36
6	wesley-Inspiron-3442	C0-38-96-A2-CE-55	192.168.0.13	01:42:17
7	android-cdc9c25c365d054e	58-3F-54-FB-3D-B0	192.168.0.12	01:14:51
8	ESP_3A5T2P	34-BB-26-CA-F3-44	192.168.0.7	01:04:24
9	ESP_4CF140	98-39-8E-43-00-2F	192.168.0.6	01:28:48
10	ESP_8DFA01	DC-BF-E9-17-2F-4D	192.168.0.10	00:22:59
11	ESP_61AEA2	60-01-94-61-AE-A2	192.168.0.11	01:32:22
12	Ubuntu-Server-TCC	00-13-EF-50-02-AF	192.168.0.8	Permanente
13	ESP_8BD5E3	68-C6-3A-8B-D5-E3	192.168.0.16	01:31:27
14	ESP_D6B7AD	5C-CF-7F-D6-B7-AD	192.168.0.17	01:33:29
15	ESP_CAE0D9	18-FE-34-CA-E0-D9	192.168.0.18	01:30:42
16	ESP_8BCEFB	68-C6-3A-8B-CE-FB	192.168.0.19	01:29:49
17	ESP_0AF610	A0-20-A6-0A-F6-10	192.168.0.20	01:28:08

fonte: o autor. Imagem obtida do painel de controle do roteador.

A figura 10, mostra a página web fictícia criada para o servidor web (Apache) exibir enquanto sua execução normal.

Figura 10: Página Web entregue pelo servidor Web



Servidor Apache de Teste

Universidade Federal do Ceará

Curso de Engenharia de Computação

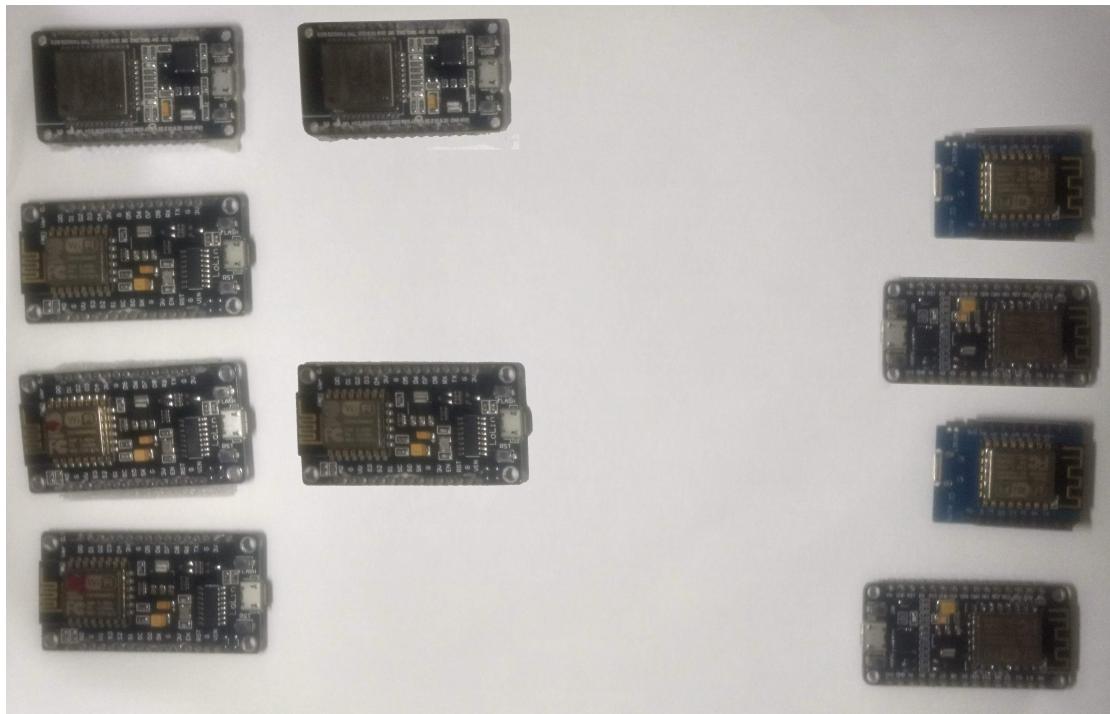
Trabalho de Conclusão de Curso

Francisco Wesley Melo Silva

fonte: o autor.

A figura 11, mostra os dispositivos utilizados para execução do ataque DDoS.

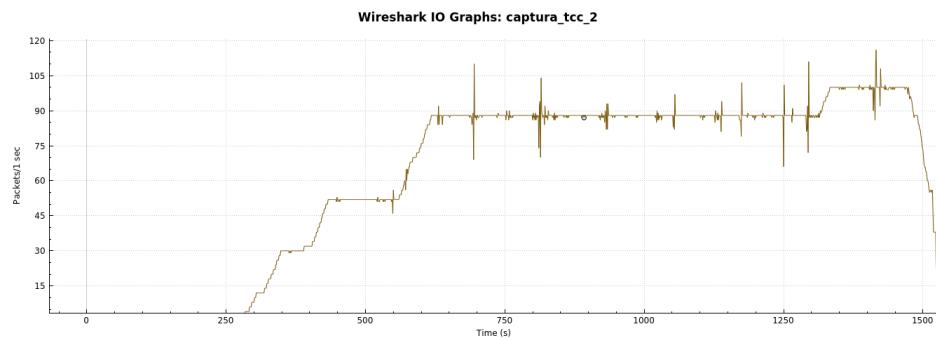
Figura 11: Dispositivos utilizados para o ataque DDoS



fonte: o autor.

A figura 12, retrata a quantidade de pacotes enviadas num curto espaço de tempo durante a execução do ataque.

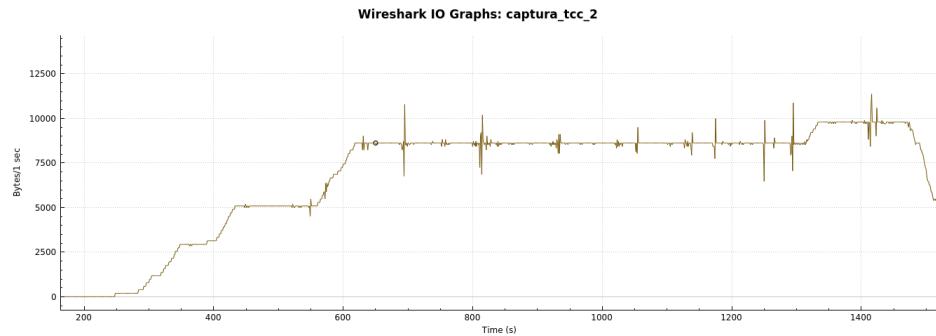
Figura 12: Fluxo de pacotes com destino ao servidor web



fonte: o autor.

A figura 13, retrata a quantidade de bytes que percorria a rede ao dispositivo atacado.

Figura 13: Fluxo de bytes com destino ao alvo



fonte: o autor.

A figura 14, a seguir foi extraída da tela de um navegador web ao tentar acessar a página web que deveria ser provida pelo servidor web.

Figura 14: Tela do navegador contendo erro



fonte: o autor.

6 Resultados obtidos

Durante a execução do experimento, foi possível identificar que o grande fluxo de dados que incorre sobre a rede em direção a um nó em específico é limitado pela capacidade de tráfego da camada física e dos nós que interceptam o fluxo de pacotes. Roteadores *wireless* convencionais tem taxa de transferência limitada à cerca de 300 mbps, suficientes para suportar o tráfego durante

o ataque. Ao direcionar requisições do tipo *echo request* para um nó específico da rede, neste caso, o computador que abrigava o servidor web, a taxa de transferência chegou a 1,28 Kilobyte por segundo, puramente de *echo requests*, incluindo os demais datagramas que encapsulam o datagrama ICMP.

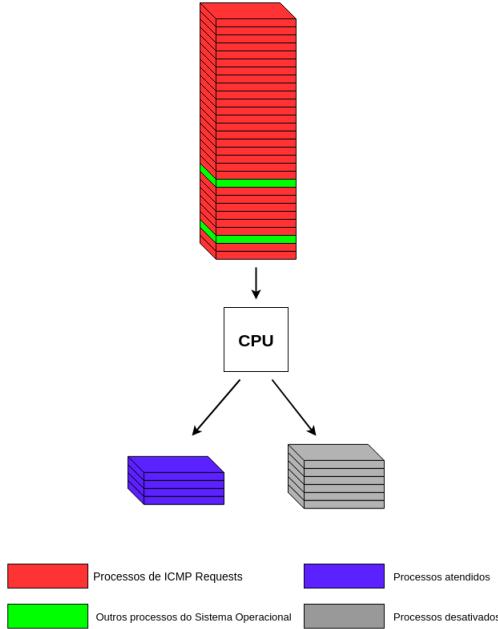
No total, foram identificados através do *Wireshark*, 125514 pacotes de dados trafegando pela rede em direção ao nó alvo do ataque, sendo 95266 do tipo *ICMP echo request*.

Como a maioria do tráfego trata-se de requisições ao nó atacado, naturalmente o alvo tentou responder à tal demanda com o máximo de eficiência possível. Porém, como esperado não foi possível atender à tantas requisições. De acordo com Tanembaum (2010, p.463), esse conjunto de requisições não correspondidas é alocado numa “fila de execução” e destinada à cada CPU (*Central Processing Unit* ou Unidade Central de Processamento).

O escalonador do sistema operacional tenta seleciona um processo de mais alta prioridade e quando um tempo limite predeterminado é atingido, os processos são movidos para uma pilha de processos expirados, mas devido a grande quantidade de processos de mesma prioridade ainda não executados, tal ação não surte o efeito de aliviar a carga de trabalho nas CPUs.

A figura 15, a seguir exemplifica o chaveamento de processos no sistema operacional durante um ataque DDoS.

Figura 15: Chaveamento de processos no sistema operacional.



fonte: o autor.

Foram identificados cerca de 37615 pacotes em resposta aos 67646 *echo requests*. O tempo entre o início do ataque e primeiros sinais de instabilidade no funcionamento da máquina que hospedara o servidor *web* foi de aproximadamente 92 minutos. Após cerca de 135 minutos de envio de requisições, a máquina passou a apresentar sinais mais visíveis de mau funcionamento, como travamentos totais eventuais. Embora ainda respondendo a algumas poucas requisições ICMP. Aproximadamente 280 minutos depois do início do experimento, a máquina parou de responder totalmente as *echo requests*.

Devido aos esforços para atender tais requisições, o sistema operacional entra em colapso ao priorizar atender tais tarefas ao invés de continuar a prover os serviços essenciais para o funcionamento do equipamento. Consequentemente, a atividade-fim de atuar como um servidor de páginas para *web* deixa ser cumprida, à princípio causando lentidão durante a entrega de páginas ao usuário. E por fim, a inoperabilidade total do serviço após 311 minutos.

Enquanto o experimento ocorria, um computador ligado a mesma rede acessava periodicamente a página web hospedada na máquina servidora.

7 Conclusão

A evolução na infraestrutura das redes de telecomunicação possibilitou que ideias cada vez mais sofisticadas para conectar a vida das pessoas à rede fossem concretizadas. Contudo, cada dispositivo inteligente conectado à Internet pode ser utilizados de modo nem sempre positivo. Usuários mal intencionado, seja o consumidor final ou mesmo os fabricantes de produtos, podem empregar à capacidade de conexão à rede e a estrutura que compõe a Internet através dos protocolos de comunicação como o ICMP para performar requisições a um dispositivo da rede.

A utilização de dispositivos IoT para ataques a redes deve ser considerada pois, devido ao baixo custo é possível a montagem de grandes malhas compostas de pequenos artefatos dotados de um microprocessador e os demais componentes que possibilitam sua conexão com a Internet.

Servidores Web são potenciais alvos, quando o gerenciamento da rede é falho de modo que um grande fluxo de requisições pode ser enviado para um único destinatário. O alvo de tal ataque, devido a suas limitações de *hardware*, é incapaz de responder à tantas requisições e ocasionalmente entra em colapso e, consequente para de prover o serviço que fora designado.

Tal falha, dependendo da função que o servidor executa, pode ter como consequência a instabilidade de sistemas conectados ao servidor, como também a inoperabilidade do total do mesmo. Como consequência, podem ser acarretados graves problemas, tanto financeiros, como pôr em risco a vida de usuários finais, por exemplo: ao causar instabilidade num servidor web que registre em tempo real sobre sinais vitais de um paciente clínico.

Mesmo os sistemas operacionais modernos, não estão preparados para lidar com esse tipo de ataque, cabendo aos responsáveis pelo gerenciamento da rede criar barreiras que dificultem esse tipo de ataque. Bloquear todas as requisições ICMP não solucionam de forma eficiente tal problema, segundo a RFC 5927 (2010, p.10) que descreve diversos ataques utilizando o protocolo ICMP, principalmente a técnica de *ICMP flooding* e como evitar tais ataques. Por exemplo, ao aplicar regras de Netfilter em sistemas GNU/Linux que estão geralmente presentes em roteadores utilizados comercialmente.

Referências

- [Asrodia e Sharma 2013]ASRODIA, P.; SHARMA, M. V. Network monitoring and analysis by packet sniffing method. *International Journal of Engineering Trends and Technology (IJETT)*, v. 4, 2013. ISSN 2231-5381.
- [Buyya 2016]BUYYA, R. *Internet of Things: Principles and Paradigms*. [S.l.]: Morgan Kaufmann, 2016. ISBN 978-0-12-805395-9.
- [daCosta e Henderson 2014]DACOSTA, F.; HENDERSON, B. *Rethinking the Internet of Things: A Scalable Approach to Connecting Everything*. [S.l.]: Apress, 2014. ISBN 978-1-4302-5741-7.
- [Espressif 2018]ESPRESSIF. *ESP32 Series Datasheet*. 2. ed. [S.l.], 2018.
- [Espressif 2018]ESPRESSIF. *ESP8266EX Datasheet*. 5. ed. [S.l.], 2018.
- [Filho 2013]FILHO, J. E. M. *Gucci Mane Poster Print 32x24inch*. [S.l.]: Novatec, 2013. ISBN 9788575223758.
- [Forouzan 2007]FOROUZAN, B. A. *Data Communications and Networking*. 4 edition. ed. [S.l.]: McGraw-Hill Higher Education, 2007. ISBN 9780072967753.
- [Gont 2010]GONT, F. *ICMP Attacks against TCP*. [S.l.], 2010. Disponível em: <<https://tools.ietf.org/html/rfc5927section-4.2>>.
- [Goyal e Goyal 2017]GOYAL, P.; GOYAL, A. Comparative study of two most popular packet sniffing tools- tcpdump and wireshark. *9 th International Conference on Computational Intelligence and Communication Networks*, 2017.
- [Gu e Liu 2007]GU, Q.; LIU, P. *Denial of Service Attacks*. 2007. 28 p. Disponível em: <<https://s2.ist.psu.edu/ist451/DDoS-Chap-Gu-June-07.pdf>>.
- [Kodali e Soratkal 2017]KODALI, R. K.; SORATKAL, S. Mqtt based home automation system using esp8266. *National Institute Of Technology*, v. 1, 2017.
- [Madson 2018]MADSON, U. of W. *EMBEDDED SYSTEM OVERVIEW*. 2018. Disponível em: <<https://ece353.engr.wisc.edu/ece353-the-book/embedded-system-overview/EmbeddedSystem>>.

[Morimoto 2013]MORIMOTO, C. E. *Servidores Linux: Guia Pratico.* [S.l.]: Sulina, 2013. ISBN 9788599593134.

[Pastor-Satorras e Vespignani 2007]PASTOR-SATORRAS, R.; VESPIGNANI, A. *Evolution and Structure of the Internet: A Statistical Physics Approach.* [S.l.]: Cambridge University Press, 2007. ISBN 9780521714778.

[Tanenbaum 2009]TANENBAUM, A. S. *Sistemas Operacionais Modernos.* [S.l.]: Pearson, 2009. ISBN 9788576052371.

[Tanenbaum e Steen 2006]TANENBAUM, A. S.; STEEN, M. V. *Distributed Systems: Principles and Paradigms (2nd Edition).* [S.l.]: Pearson, 2006. ISBN 0132392275.

Apêndice

Código-fonte utilizado

Algorithm 2 Algoritmo utilizado no ataque

```
1  \#include <ESP826WiFi.h>
2
3  \#include <ESP8266Ping.h>
4
5
6  const char{*} ssid = \textquotedbl\#\#\#\#\#\textquotedbl ; //  
7  SSID da rede que será conectado
8
9  const char{*} password = \textquotedbl\#\#\#\#\#\textquotedbl ;
10 // Senha da rede que será conectada
11
12 const char{*} remote\_host = \textquotedbl\#\#\#\.\#\#\#\#\#\#\#\#\#\#\textquotedbl ;
13 //Endereço IP ou URL do dispositivo alvo
14
15 void setup() \{
16
17 Serial.begin(115200); // Inicia a comunicação serial
18
19 delay(10); // Aguarda 10 milissegundos
20
21 // Inicia a conexão à rede sem-fio.
22
23 Serial.println();
24
25 Serial.println(\textquotedbl Conectando à rede Wi-Fi\textquotedbl );
26
27 WiFi.begin(ssid, password);
28
29 while (WiFi.status() != WL\_CONNECTED) \{ // verifica se conectado
30 ou não.
31
32 delay(100);
33
34 Serial.print(\textquotedbl .\textquotedbl ); \}
35
36 Serial.println();
37
38 Serial.print(\textquotedbl WiFi conectado com IP \textquotedbl );
39 //Exibe o endereço local do dispositivo
40
41 Serial.println(WiFi.localIP());\}
42
43 void loop() \{
44
45 Serial.print(\textquotedbl Envio de ping ao host \textquotedbl );
46
47 Serial.println(remote\_host);
48
49 if(Ping.ping(remote\_host)) \{ //Verifica resposta de ping contra
50 o objeto atacado
51
52 Serial.println(\textquotedbl Sucesso!!\textquotedbl ); \} else \{
53
54 Serial.println(\textquotedbl Erro!:(\textquotedbl ); \}
55
56 delay(500); \} //Aguarda meio segundo e refazer ICMP \emph{echo}.
```