

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Suelen Cristina Cremonini

**Internet das Coisas e seu Potencial para
Ataques de Negação de Serviço**

Uberlândia, Brasil

2017

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Suelen Cristina Cremonini

**Internet das Coisas e seu Potencial para Ataques de
Negação de Serviço**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau
de Bacharel em Ciência da Computação.

Orientador: Rodrigo Sanches Miani

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2017

Suelen Cristina Cremonini

Internet das Coisas e seu Potencial para Ataques de Negação de Serviço

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 22 de dezembro de 2017:

Rodrigo Sanches Miani
Orientador

Paulo Henrique Ribeiro Gabriel
Convidado 1

Maria Adriana Vidigal de Lima
Convidado 2

Uberlândia, Brasil
2017

Resumo

A cada dia a tecnologia evolui, tornando-se mais rápida e menor e movendo-se em direção ao modelo “sempre conectado”. Essa evolução favorece a comunicação entre computadores e cria um novo conceito de Internet para o futuro: a Internet das Coisas (do inglês *Internet of Things* - IoT). Junto a essa evolução, entretanto, vem também uma grande preocupação com vulnerabilidades nos dispositivos da IoT. Uma das formas de explorar as vulnerabilidades na IoT é com um ataque conhecido como DoS (do inglês *Denial of Service*) a partir dos dispositivos nela presentes. A fim de verificar o potencial da IoT para tais ataques, simulações utilizando o simulador Cooja foram propostas neste trabalho. Após implementar os ataques, experimentos foram realizados e com a análise de seus resultados concluiu-se que é possível criar tais ataques e verificar sua eficácia por meio do atraso na coleta de informações pelo sensor sendo atacado.

Palavras-chave: Internet das Coisas, Segurança da Informação, Ataques, Negação de Serviço, Simulação.

Lista de ilustrações

Figura 1 – Rede utilizada no trabalho de Hengst (2016)	18
Figura 2 – Pilha de protocolos utilizada no trabalho de Pacheco et al. (2016)	19
Figura 3 – Exemplo de DODAG. Retirada de Sheng, Yang e Yu (2013)	22
Figura 4 – Níveis de abstração do NS-2, TOSSIM, Avrora e Cooja. Retirada de Österlind et al. (2006)	24
Figura 5 – Tela inicial do Cooja.	24
Figura 6 – Nova simulação.	25
Figura 7 – Rede WSN com 5 nós.	25
Figura 8 – Rede com um nó atacante e um nó comum. Retirada de Zhang e Green (2015).	27
Figura 9 – Rede com um nó legítimo e um nó comum. Retirada de Zhang e Green (2015).	27
Figura 10 – Rede com um nó legítimo, um nó comum e múltiplos nós atacantes. Retirada de Zhang e Green (2015).	27
Figura 11 – Lista de cenários utilizados no trabalho de Borgiani et al. (2017). . . .	28
Figura 12 – Lista de parâmetros utilizados nas simulações do trabalho de Borgiani et al. (2017).	28
Figura 13 – Código utilizado para simular o ataque de negação de serviço.	29
Figura 14 – Rede composta de 1 alvo (nó 2) e 1 atacante (nó 1).	30
Figura 15 – Rede composta de 1 alvo (nó 1), 1 atacante (nó 2) e 1 outro nó.	30
Figura 16 – Rede composta de 1 alvo (nó 7), 1 atacante (nó 2) e 10 outros nós. . . .	30
Figura 17 – Rede composta de 1 alvo (nó 1) e 5 atacantes (nós 2, 3, 4, 5 e 6). . . .	30
Figura 18 – Rede composta de 1 alvo (nó 1), 5 atacantes (nós 2, 3, 4, 5 e 6) e 6 outros nós.	30
Figura 19 – Dispositivo real emulado pelo <i>tmote sky</i> . Retirado de IV (2017).	31
Figura 20 – Cenário 1: Battery Voltage antes e durante o ataque	34
Figura 21 – Cenário 1: Light 1 antes e durante o ataque	34
Figura 22 – Cenário 1: Light 2 antes e durante o ataque	34
Figura 23 – Cenário 2: Battery Voltage antes e durante o ataque	35
Figura 24 – Cenário 2: Light 1 antes e durante o ataque	35
Figura 25 – Cenário 2: Light 2 antes e durante o ataque	35
Figura 26 – Cenário 3: Battery Voltage antes e durante o ataque	36
Figura 27 – Cenário 3: Light 1 antes e durante o ataque	36
Figura 28 – Cenário 3: Light 2 antes e durante o ataque	36
Figura 29 – Cenário 4: Battery Voltage antes e durante o ataque	37
Figura 30 – Cenário 4: Light 1 antes e durante o ataque	37

Figura 31 – Cenário 4: Light 2 antes e durante o ataque	37
Figura 32 – Cenário 5: Battery Voltage antes e durante o ataque	38
Figura 33 – Cenário 5: Light 1 antes e durante o ataque	38
Figura 34 – Cenário 5: Light 2 antes e durante o ataque	38
Figura 35 – Execução 2 - Cenário 1: Battery Voltage antes e durante o ataque . . .	57
Figura 36 – Execução 2 - Cenário 1: Light 1 antes e durante o ataque	58
Figura 37 – Execução 2 - Cenário 1: Light 2 antes e durante o ataque	58
Figura 38 – Execução 2 - Cenário 2: Battery Voltage antes e durante o ataque . . .	58
Figura 39 – Execução 2 - Cenário 2: Light 1 antes e durante o ataque	59
Figura 40 – Execução 2 - Cenário 2: Light 2 antes e durante o ataque	59
Figura 41 – Execução 2 - Cenário 3: Battery Voltage antes e durante o ataque . . .	59
Figura 42 – Execução 2 - Cenário 3: Light 1 antes e durante o ataque	60
Figura 43 – Execução 2 - Cenário 3: Light 2 antes e durante o ataque	60
Figura 44 – Execução 2 - Cenário 4: Battery Voltage antes e durante o ataque . . .	60
Figura 45 – Execução 2 - Cenário 4: Light 1 antes e durante o ataque	61
Figura 46 – Execução 2 - Cenário 4: Light 2 antes e durante o ataque	61
Figura 47 – Execução 2 - Cenário 5: Battery Voltage antes e durante o ataque . . .	61
Figura 48 – Execução 2 - Cenário 5: Light 1 antes e durante o ataque	62
Figura 49 – Execução 2 - Cenário 5: Light 2 antes e durante o ataque	62
Figura 50 – Execução 3 - Cenário 1: Battery Voltage antes e durante o ataque . . .	62
Figura 51 – Execução 3 - Cenário 1: Light 1 antes e durante o ataque	63
Figura 52 – Execução 3 - Cenário 1: Light 2 antes e durante o ataque	63
Figura 53 – Execução 3 - Cenário 2: Battery Voltage antes e durante o ataque . . .	63
Figura 54 – Execução 3 - Cenário 2: Light 1 antes e durante o ataque	64
Figura 55 – Execução 3 - Cenário 2: Light 2 antes e durante o ataque	64
Figura 56 – Execução 3 - Cenário 3: Battery Voltage antes e durante o ataque . . .	64
Figura 57 – Execução 3 - Cenário 3: Light 1 antes e durante o ataque	65
Figura 58 – Execução 3 - Cenário 3: Light 2 antes e durante o ataque	65
Figura 59 – Execução 3 - Cenário 4: Battery Voltage antes e durante o ataque . . .	65
Figura 60 – Execução 3 - Cenário 4: Light 1 antes e durante o ataque	66
Figura 61 – Execução 3 - Cenário 4: Light 2 antes e durante o ataque	66
Figura 62 – Execução 3 - Cenário 5: Battery Voltage antes e durante o ataque . . .	66
Figura 63 – Execução 3 - Cenário 5: Light 1 antes e durante o ataque	67
Figura 64 – Execução 3 - Cenário 5: Light 2 antes e durante o ataque	67

Lista de tabelas

Tabela 1 – Cenários utilizados nas simulações	31
Tabela 2 – Tempo gasto para simular cada cenário antes e durante o ataque e tempo total da simulação para cada cenário	31
Tabela 3 – Parâmetros utilizados nas simulações	32
Tabela 4 – Frequência de envio de pacotes <i>Hello</i>	39

Lista de abreviaturas e siglas

IoT	Internet of Things
DDoS	Distributed Denial of Service
DoS	Denial of Service
DNS	Domain Name System
IDS	Intrusion Detection System
DD4BC	DDoS for Bitcoin
TI	Tecnologia da Informação
WSN	Wireless Sensor Network
CoAP	Constrained Application Protocol
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
RPL	Routing Protocol for Low Power and Lossy Networks
6LowPAN	IPv6 over Low Power Wireless Personal Area Networks
MAC	Medium Access Control
PHY	Physical Layer
MTU	Maximum Transmission Unit
DODAG	Destination-oriented acyclic graph
SO	Sistem Operacional

Sumário

1	INTRODUÇÃO	9
2	REVISÃO BIBLIOGRÁFICA	11
2.1	Segurança da Informação	11
2.2	Ataque de Negação de Serviço	12
2.3	Internet das Coisas	14
2.4	Trabalhos Correlatos	16
3	DESCRIÇÃO DO AMBIENTE COOJA/CONTIKI	20
3.1	Arquitetura IoT	20
3.2	Contiki	22
3.3	Cooja	23
4	ATAQUES DOS USANDO O COOJA	26
4.1	Experimentos	29
4.2	Resultados	32
5	CONCLUSÃO	40
	Conclusão	40
	REFERÊNCIAS	41
	APÊNDICES	44
	APÊNDICE A – CÓDIGO DO ATAQUE	45
	APÊNDICE B – SIMULAÇÕES EXTRAS DOS CENÁRIOS UTILIZADOS	57

1 Introdução

A cada dia a tecnologia evolui, tornando-se mais rápida e menor e movendo-se em direção ao modelo “sempre conectado”. Essa evolução tecnológica observada durante as primeiras décadas do século XXI favorece a comunicação entre computadores e cria um novo conceito de Internet para o futuro: a Internet das Coisas (do inglês *Internet of Things* - IoT). Na IoT, “coisas” presentes no cotidiano (como cafeteiras, relógios e despertadores) são conectadas à Internet de forma que informações críticas podem ser enviadas através da rede para qualquer lugar a qualquer momento (SONAR; UPADHYAY, 2014).

Como a IoT é um paradigma que facilita a troca de informações, ela é usada para suportar serviços globais e redes de fornecimento de bens em diversas áreas como indústria, logística, educação e saúde (ZHANG; GREEN, 2015). Graças a este suporte, novos modelos de negócios tornam-se possíveis em toda a indústria. Estes modelos oferecem redução de custos e de riscos, aprimoramento no processo empresarial e ainda criam grandes oportunidades no meio corporativo, de forma a mudar o rumo e o ritmo dos negócios (RAYES; SALAM, 2017). Além de, segundo Zhang e Green (2015), ter sido considerada como a terceira onda de tecnologia da informação após Internet e a rede móvel, Rayes e Salam (2017) apontam que especialistas concordam que a Internet das coisas revolucionará as empresas além do imaginável nas próximas décadas.

Entretanto, junto a essa revolução da tecnologia vem também uma grande preocupação com vulnerabilidades nos dispositivos da IoT. Problemas como privacidade, controle de acesso, configuração do sistema, armazenamento de informações e gerenciamento são os principais desafios neste ambiente. Logo, para adotar o uso extensivo da IoT, os problemas de segurança devem antes ser solucionados (ALABA et al., 2017).

Uma das formas de explorar as vulnerabilidades na IoT é com um ataque conhecido como DDoS (do inglês *Distributed Denial of Service*) a partir dos dispositivos nela presentes. Neste ataque, uma rede de máquinas comprometidas envia requisições simultâneas a um alvo com a finalidade de sobrecarregar seu sistema e impedir que usuários legítimos usem seus recursos. Em outubro de 2016 um ataque do tipo DDoS interferiu com o acesso a vários serviços populares como *Twitter*, *Amazon* e *Tumblr* e os principais participantes do ataque foram dispositivos da IoT (ANGRISHI, 2017).

Pacheco et al. (2016) e Hengst (2016) a fim de estudarem, respectivamente, a viabilidade e a potencialidade de ataques de DDoS em ambiente IoT, simularam tais ataques utilizando, respectivamente, o simulador NS-3 e uma rede real de dispositivos conectados. Para o primeiro trabalho, os resultados apontaram que o uso de IoT de fato é viável, o que validou a preocupação dos autores com a questão da segurança. No segundo

trabalho, a autora indicou que em 2020 a intensidade de um ataque DDoS utilizando a IoT deve ser de pelo menos 36,28 TBps, um número 30 vezes maior que o ataque de outubro de 2016, o qual, segundo [Mansfield-Devine \(2016\)](#), chegou a 1,2 TBps e atualmente é o ataque com maior volume conhecido.

O objetivo deste trabalho é verificar o potencial da IoT para ataques de negação de serviço. Para isso, experimentos utilizando o simulador Cooja – que é um simulador criado para simular redes no ambiente da Internet das Coisas – foram criados. Nesses experimentos, ataques de negação de serviço são lançados a partir de dispositivos IoT com outros dispositivos IoT como alvos e uma análise é feita a fim de validar a eficácia do ataque recebido por esses dispositivos. Os resultados dessa análise apontam que é possível criar tais ataques e ainda verificar sua efetividade por meio do atraso na coleta de informações pelo sensor sendo atacado.

Este trabalho está dividido da seguinte forma: o capítulo 2 aborda conceitos básicos sobre Segurança da Informação - seus pilares, algumas vulnerabilidades que podem ser exploradas e formas de evitar essa exploração, também discute sobre Ataques de Negação de Serviço - suas classificações, formas de detecção e contramedidas, trata da Internet das Coisas - sua definição, objetivo, vulnerabilidades existentes e formas de proteger os dispositivos pertencentes a ela, e por último, apresenta os trabalhos correlacionados. O capítulo 3 descreve a arquitetura IoT - seus protocolos e a padronização destes, o ContikiSO - um sistema operacional criado para dispositivos IoT, e o Cooja - um simulador de redes desenvolvido exclusivamente para redes IoT. O capítulo 4 traz alguns exemplos de simulações de ataques DoS/DDoS utilizando o Cooja, apresenta os experimentos implementados nesta monografia e seus resultados. Por fim, a seção [Conclusão](#) apresenta os resultados obtidos e os trabalhos futuros.

2 Revisão Bibliográfica

2.1 Segurança da Informação

Informação é de importância vital para muitas organizações. Por conseguinte, precisa ser protegida, assegurada e gerida adequadamente. Se, por qualquer razão, ela for comprometida, a organização pode perder tempo, mão de obra, dinheiro e/ou oportunidades de negócios (KRITZINGER; SMITH, 2008). Sua proteção é chamada de “segurança da informação” e, segundo Bishop (2002), é frequentemente definida em termos de:

- Confidencialidade: a informação deve ser acessível somente a quem possui autorização do proprietário da informação;
- Disponibilidade: a informação deve estar disponível sempre que requisitada por usuários autorizados;
- Integridade: mudanças impróprias ou sem a autorização do proprietário da informação devem ser impedidas.

Segurança da informação é um problema crítico que afeta organizações por todo o mundo (KNAPP et al., 2006). Seu principal objetivo é o de proteger a informação e garantir que a sua disponibilidade, confidencialidade e integridade não sejam comprometidas (KRITZINGER; SMITH, 2008). Contudo, com a inovação e o desenvolvimento que o século 21 nos trouxe, veio também uma forte dependência em infraestrutura de TI (DLAMINI; ELOFF; ELOFF, 2009). Enquanto na década de 1990 a maioria dos *hackers* eram inexperientes e motivados principalmente por curiosidade ou pela busca de fama, hoje eles são profissionais que procuram por ganhos financeiros, realizam ativismo político ou estão envolvidos em espionagem cibernética patrocinada pelo estado (TRAORE; AWAD; WOUNGANG, 2017).

No trabalho de Miani (2013), há a citação das seguintes falhas que, se exploradas corretamente por esses atacantes, podem comprometer a segurança do sistema:

- Exploração de vulnerabilidades em sistemas operacionais, aplicativos, protocolos e serviços. Exemplo: execução remota de códigos maliciosos em sistemas operacionais como o Windows;
- Exploração dos aspectos humanos dos envolvidos. Exemplo: uso de engenharia social para a obtenção de informações sigilosas;

- Falha no desenvolvimento e implementação da política de segurança. Exemplo: elaboração de política de segurança que não prevê o uso de cópias de segurança;
- Falha na configuração de serviços e de sistemas de segurança. Exemplo: utilização da configuração padrão que é conhecida por todos, inclusive pelos invasores.

A fim de auxiliar no combate a ataques, [Nakamura e Geus \(2010\)](#) descreveram as seguintes ferramentas a serem utilizadas:

- *Firewall*: é um ponto entre duas ou mais redes, que pode ser um componente ou um conjunto de componentes, por onde passa todo o tráfego. Utilizando este ponto, é possível controlar, autenticar e registrar o tráfego que passa por ele. Além disso, também pode ser utilizado para restringir o acesso de uma rede a outra;
- Anti-vírus: atua na detecção de vírus, *worms* e cavalos de Troia. Os *worms* diferem-se dos vírus por espalharem-se rápida e automaticamente, sem a necessidade de interação com o usuário, como ocorre com os vírus. Os cavalos de Troia são programas que aparentam realizar alguma tarefa útil, mas, na verdade, realizam atividades prejudiciais.
- Sistemas de detecção de intrusão: sistema cujo objetivo é o de detectar atividades suspeitas, impróprias, incorretas ou anômalas. É capaz de detectar possíveis ataques tanto internos quanto externos. Externamente, pode detectar tanto ataques que seriam normalmente barrados pelo *firewall* quanto ataques realizados por meio de portas legítimas (que não são protegidas por *firewall*).
- Políticas de segurança: são políticas definidas por uma determinada organização nas quais estão estabelecidas as precauções que devem ser tomadas ao manusear informações pertinentes àquela organização e as punições e procedimentos a serem seguidos no caso do seu não-cumprimento.

2.2 Ataque de Negação de Serviço

Uma forma de ataque que explora, entre outras, a falha na configuração de serviços e de sistemas de segurança é o ataque de negação de serviço (do inglês *denial of service* - DoS). Este ataque pode ser caracterizado como um ataque cujo objetivo é impedir que usuários legítimos usem alguns recursos da rede, tais como *sites*, *web services* e sistemas computacionais ([CUSACK; TIAN; KYAW, 2017](#)), sendo ele, portanto, um ataque que afeta a disponibilidade da informação.

[Feitosa, Souto e Sadok \(2008\)](#) e [Laufer et al. \(2005\)](#) mencionam a seguinte classificação de ataques DoS:

- Ataques por inundação (*flooding*): exploram o processo de estabelecimento de conexão do protocolo de transporte TCP (*three-way handshake*);
- Ataques por refletor: outro tipo de ataque de inundação. Neste, entretanto, existe uma estação intermediária (refletor) entre a vítima e o atacante;
- Ataques à infraestrutura de rede: o ataque ocorre em algum ponto crucial para o funcionamento do serviço, sendo que tal ponto não depende da vítima. Um exemplo é o ataque ao servidor DNS;
- Ataques por vulnerabilidade: explora alguma vulnerabilidade na pilha de protocolos ou na própria aplicação;
- Ataques distribuídos: geralmente utilizados em ataques de inundação quando uma estação sozinha não é capaz de afetar a disponibilidade da vítima.

Com mencionado acima, uma das classificações do ataque DoS é o ataque distribuído, também conhecido como DDoS. No ataque DDoS, um hacker escraviza temporariamente uma série de dispositivos conectados à Internet combinando-os em um arranjo conhecido como *botnet* (ANGRISHI, 2017).

Botnet é uma rede de máquinas comprometidas, também conhecidas como *bots*, que executam softwares mal intencionados sob o comando e controle de um *botmaster*. Os *bots* podem explorar automaticamente a rede e, ao se deparar com vulnerabilidades em outras máquinas, infectá-las. Uma vez que uma máquina está comprometida, um programa é instalado para futura ativação pelo *botmaster*, que em determinado momento pode instruir os *bots* na rede a executar determinadas ações (BERTINO, 2017), tais como o envio de solicitações simultâneas de um serviço específico para um ou mais servidores, o que acarreta no sobrecarregamento do servidor sendo atacado e este passa a ignorar solicitações legítimas de usuários finais, resultando em um ataque DDoS (ANGRISHI, 2017).

A fim de detectar um ataque de negação de serviço, pode-se utilizar uma ferramenta denominada IDS (*Intrusion Detection System*). O IDS monitora os eventos que ocorrem em uma rede ou sistema computacional a fim de identificar possíveis invasões - tentativas de comprometer a confidencialidade e/ou a disponibilidade da informação ou de burlar mecanismos de segurança da rede ou computador. Os IDSs são implementados com métodos geralmente baseados em detecção de mau uso (*misuse-based*) ou detecção de anomalia (*anomaly-based*). Os IDSs baseados em mau uso reconhecem os ataques através de padrões, ou seja, regras e assinaturas, enquanto os ataques baseados em anomalia precisam ser treinados utilizando-se do ambiente no qual serão executados (PEREIRA, 2011).

No trabalho de [Laufer et al. \(2005\)](#), as contramedidas ao ataque de negação de serviço são divididas em medidas preventivas e medidas reativas. Como medida preventiva, a primeira e mais imediata é a manutenção de todos os softwares do sistema, mantendo-os sempre atualizados. A segunda é uma técnica denominada filtragem de ingresso (*ingress filtering*), cujo objetivo é evitar que pacotes com endereço IP forjado trafeguem na rede. Como medida reativa, é proposto o rastreamento dos pacotes a fim de identificar o seu emissor, interrompendo assim o ataque através da filtragem de pacotes.

O DDoS é um dos ataques cibernéticos mais interessantes e amplamente vistos nos últimos tempos ([ANGRISHI, 2017](#)). Um exemplo de seu uso para fins de ganho financeiro ocorreu em 2014, quando um grupo chamado DD4BC (que aparentemente significava “*DDoS for Bitcoin*”) surgiu. Este grupo visou um grande número de empresas. Inicialmente estas empresas eram de jogos online e câmbio monetário, depois passaram a ser empresas de serviços financeiros. A ideia era a seguinte: o grupo lançaria um pequeno ataque (10-20Gbps) como prova de suas habilidades. Em seguida, exigiria um pagamento em Bitcoin, com valores variando de 10BTC até 200BTC (equivalente, na época, a US\$4.000 até US\$80.000) para não lançar um ataque de maior proporção. O grupo dizia ser capaz de lançar ataques de 400-500Gbps, mas rapidamente ficou claro que ele não tinha tamanha capacidade ([MANSFIELD-DEVINE, 2016](#)).

Além da motivação financeira, ataques DDoS podem ser usados para prejudicar a imagem da vítima. Como este é um ataque visível para a comunidade externa - o usuário tenta acessar o *site* da organização mas não obtém sucesso e relata isso via *Twitter*, por exemplo -, isso causa à empresa um certo impacto em sua reputação. Outra motivação envolve motivos políticos - como em um ataque ocorrido em 2007, quando *hackers* russos prejudicaram grande parte da infraestrutura digital da Estônia. Ataques DDoS também podem ser utilizados para mascarar outros tipos de ataques. Nesse contexto, são lançados múltiplos ataques com a intenção de confundir a equipe de TI (Tecnologia da Informação) da vítima. Enquanto a equipe está ocupada, os atacantes podem montar outros tipos de ataques, como por exemplo o roubo de dados cruciais da organização ([MANSFIELD-DEVINE, 2016](#)).

2.3 Internet das Coisas

A Internet das Coisas refere-se, de um modo geral, à interconexão em rede de objetos presentes no cotidiano, sendo que estes objetos são, muitas vezes, equipados com inteligência ubíqua. A IoT aumentará a ubiquidade da Internet através da integração destes objetos a fim de fazê-los interagir através de sistemas embarcados, o que leva a uma rede altamente distribuída de dispositivos que se comunicam com seres humanos e outros dispositivos ([XIA et al., 2012](#)). Assim, o principal objetivo da IoT é fornecer uma

infra-estrutura de rede com protocolos e softwares que permitam a conexão e incorporação de computadores pessoais, sensores físicos/virtuais, dispositivos inteligentes, automóveis e "coisas", tais como geladeira, microondas, alimentos e medicamentos, a qualquer hora e em qualquer rede (ALABA et al., 2017).

Em 2016, existiam 9 bilhões de dispositivos inteligentes (à exceção de *smartphones*, *tablets* e computadores) e acredita-se que em 2020 este número subirá para 28,1 bilhões (ANGRISHI, 2017). Visto que o número de dispositivos conectados à Internet aumentará drasticamente, há uma preocupação com possíveis ameaças à segurança dos usuários destes dispositivos (PACHECO et al., 2016), afinal as consequências de ataques bem-sucedidos poderiam impactar na vida e na segurança das vítimas, além de, direta ou indiretamente, causar destruição e morte (LINDQVIST; NEUMANN, 2017).

Ao contrário de computadores em geral, dispositivos IoT estão mais próximos dos usuários. Mesmo que até então existam poucos casos conhecidos nos quais houve dano intencional - na década de 1960, por exemplo, quando programas causaram a autodestruição de discos rígidos, ou entre 2007 e 2010 quando um *worm* conhecido como Stuxnet que foi supostamente projetado para danificar centrífugas de enriquecimento nuclear aparentemente foi bem sucedido -, é provável que esse seja um sério risco para os usuários da IoT (LINDQVIST; NEUMANN, 2017).

Se veículos ou dispositivos médicos forem remotamente controlados por atacantes, estes podem deliberadamente causar à vítima do ataque ferimentos ou até a morte. Outro risco inclui a manipulação de sensores ou a edição de informações que podem indiretamente causar outros riscos à saúde, tais como despejo de produtos químicos, interrupção de sistemas elétricos ou mudanças na rota de veículos. Uma diferença crítica entre dispositivos IoT e computadores em geral envolve seu gerenciamento. Enquanto sistemas operacionais e aplicações para computadores em geral tendem a ser atualizados facilmente, para muitos dispositivos IoT isso tende a ser difícil ou até impossível, afinal muitos destes dispositivos serão utilizados até que sejam descartados, recolhidos ou esquecidos (LINDQVIST; NEUMANN, 2017). Assim, a distinção crucial entre dispositivos IoT e computadores em geral é que a segurança nos dispositivos é precária, isso quando há alguma (ANGRISHI, 2017).

Um ataque DDoS ocorrido em outubro de 2016 demonstrou as possibilidades existentes ao explorar as vulnerabilidades dos dispositivos IoT. Este ataque utilizou um *malware* conhecido como Mirai que procura por dispositivos IoT vulneráveis conectados à Internet e que teve seu código fonte publicado gratuitamente. Neste ataque, o Mirai explorou especificamente senhas codificadas para serviços Telnet/SSH que os usuários não podiam desativar e foi capaz de encontrar e infectar dezenas de milhões de computadores, incluindo principalmente dispositivos como câmeras de segurança, roteadores, e talvez até TVs e gravadores de vídeo. O ataque foi direcionado aos serviços DNS oferecidos pela em-

presa Dyn, interferindo com a disponibilidade dos serviços de grandes *sites* como *Twitter*, *Amazon*, *Tumblr*, *Reddit*, *Spotify* e *Netflix* (LINDQVIST; NEUMANN, 2017).

Em 2015, a simulação de um ataque provou que ataques DDoS não são os únicos possíveis no ambiente IoT. Nessa simulação, alguns recursos de veículos que possuem vulnerabilidades em sua segurança foram remotamente controlados. Os hackers Charlie Miller e Chris Valasek em conjunto com Andy Greenberg simularam um ataque em um *Jeep Cherokee* que estava sendo dirigido por Greenberg em uma rodovia nos EUA. Os hackers iniciaram o ataque simplesmente modificando a temperatura do ar-condicionado, trocando a estação do rádio e ligando o para-brisa e finalizaram ao parar completamente o funcionamento do acelerador. Trocando a rodovia por um terreno vazio, o grupo continuou com a simulação e provou que além de parar o funcionamento do acelerador, também era possível parar completamente o funcionamento do freio e acessar as informações do GPS do carro, podendo rastrear a localização do veículo, medir sua velocidade e até traçar sua rota (GREENBERG, 2015).

2.4 Trabalhos Correlatos

No trabalho de Lindqvist e Neumann (2017), os autores fazem uma síntese de vários tipos de falhas na segurança de dispositivos IoT tais como senhas fixas e chaves de criptografia padrão. Além disso, também citam alguns exemplos onde o uso de dispositivos poderia trazer riscos aos usuários, tais como:

- Hospitais e estabelecimentos de saúde que tendem a usar dispositivos que já são remotamente controlados ou acessíveis via Internet como monitores de pacientes e desfibriladores;
- Alguns setores de infraestrutura como energia elétrica, petróleo e gás natural que usam dispositivos como sensores para automação, gerenciamento e controle remoto, sendo que este último pode ser acessível via Internet;
- Automóveis auto-dirigidos e interconectados entre si. Recentemente foi demonstrado que é possível assumir o controle de recursos críticos presentes nestes automóveis;
- Em dispositivos que possuem bateria existe a possibilidade de, remotamente, sobre-aquecer estes dispositivos o suficiente para causar incêndios ou explosões.

Este trabalho também exemplifica algumas medidas que podem ser utilizadas para melhorar a segurança dos dispositivos IoT, tais como:

- Os protocolos de conexão não devem ser tão simplistas e inseguros como são atualmente;

- Redes locais dentro de uma casa ou empresa devem ser adequadamente isoladas da Internet e de outras conexões externas, exceto quando as interações são desejadas e uma proteção adequada pode ser assegurada;
- Senhas fixas e chaves de criptografia padrão devem ser evitadas;
- No caso em que atualizações confiáveis não podem ser efetuadas, o recolhimento dos dispositivos deve ser feito. Para incentivar-lo, *firewalls* podem ter que reconhecer o tráfego de dispositivos a serem recolhidos e/ou comprometidos e bloquear a sua comunicação com a Internet;
- Também deve-se considerar a criação de códigos de proteção ao consumidor, regulamentação do uso desses dispositivos e punições para aqueles que a violarem.

Já no trabalho de [Hengst \(2016\)](#), a autora procura determinar a possível intensidade de um ataque DDoS realizado com o uso de dispositivos IoT como refletores. Para este fim, 3 perguntas foram respondidas:

1. Quais são os ataques DDoS mais comuns?
2. Em quais ataques DDoS dispositivos IoT podem ser usados como refletores?
3. Qual a largura de banda e o número de dispositivos necessários para lançar um ataque?

Para responder à primeira pergunta, a autora fez um estudo da literatura e encontrou os 12 tipos de ataques mais comuns. Para responder à segunda pergunta, uma rede (como visto na Figura 1) composta de uma televisão, um roteador, um *notebook* e um computador foi criada. Após analisar os 12 ataques considerados na primeira pergunta e determinar quais ataques DDoS poderiam utilizar dispositivos IoT, vários pacotes diferentes foram enviados aos dispositivos a fim de analisar sua resposta a eles. Os dispositivos responderam aos pacotes, assim, concluiu-se que os dispositivos podem ser usados em ataques DDoS. A terceira pergunta envolveu a análise de outros fatores que poderiam influenciar na intensidade do ataque em questão. Os fatores considerados foram largura de banda e número de dispositivos infectados. Considerando estatísticas de diferentes países entre 2007 e 2016, a largura de banda esperada em 2020 é de 15,85 Mbs. Já sobre o número de dispositivos que podem ser empregados em tal ataque, a autora concluiu que a quantidade esperada no ano de 2020 está entre 2,4 milhões e 200 bilhões. Assim, ao combinar o número de dispositivos com a velocidade de conexão, a autora averiguou que em 2020 a intensidade de um ataque DDoS utilizando dispositivos IoT é de pelo menos de 36,28 Tbps.

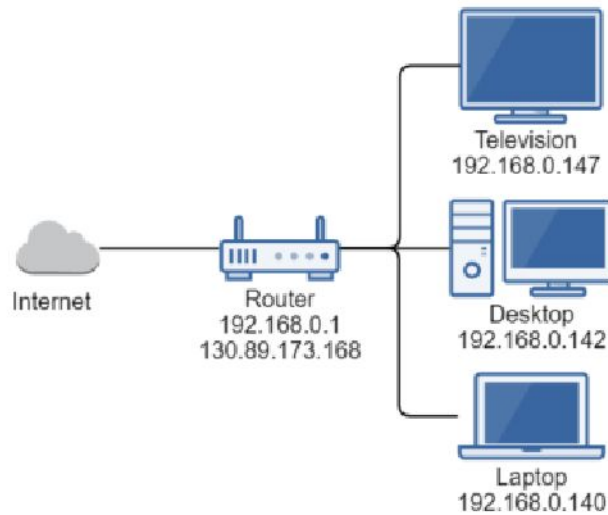


Figura 1 – Rede utilizada no trabalho de [Hengst \(2016\)](#)

No trabalho de [Pacheco et al. \(2016\)](#), os autores apresentaram um estudo da viabilidade do uso de dispositivos IoT em um ataque DDoS por reflexão amplificada. A fim de estudar tal viabilidade, os pesquisadores criaram um cenário composto por uma rede WSN (*Wireless Sensor Network*) onde há um nó infiltrado que envia requisições de descoberta de serviço para todos os sensores com o endereço de origem da vítima. Cada sensor presente nessa rede - 5 versões da rede foram criadas, cada uma possuindo, respectivamente, 5, 25, 50, 75 e 100 sensores - foi criado seguindo a pilha de protocolos de rede mostrada na figura 2. O simulador de redes NS-3 foi utilizado e uma simulação que explora a natureza do *resource discovery* do protocolo CoAP (*Constrained Application Protocol*) foi realizada.

Após analisar os resultados dessa simulação, os autores concluíram que a reflexão de fato ocorre com a amplificação, o que valida a preocupação inicial com o uso de dispositivos IoT em ataques DDoS. Contudo, o valor máximo alcançado na amplificação, para mais de 5 nós foi de 20, que é um valor muito abaixo do esperado de 76,25 para 25 nós. Assim, considerando esses resultados, os autores concluíram que para um ataque DDoS real e bem sucedido usando WSN, é necessário usar um número maior de redes.

Ao comparar os trabalhos de [Lindqvist e Neumann \(2017\)](#), [Hengst \(2016\)](#) e [Pacheco et al. \(2016\)](#) nota-se certa semelhança com o trabalho aqui desenvolvido. O trabalho de [Lindqvist e Neumann \(2017\)](#) nos dá, de certa forma, uma motivação para esta monografia visto que ele expõe várias vulnerabilidades presentes em dispositivos da IoT e cita possíveis ataques que podem usufruir destas vulnerabilidades. No trabalho de [Hengst \(2016\)](#), houve a criação de uma rede real de dispositivos IoT e o lançamento de ataques DDoS a partir destes dispositivos. Já no trabalho de [Pacheco et al. \(2016\)](#), ao invés de uma rede real, uma simulação utilizando o simulador NS-3 foi criada e ataques também foram lançados.

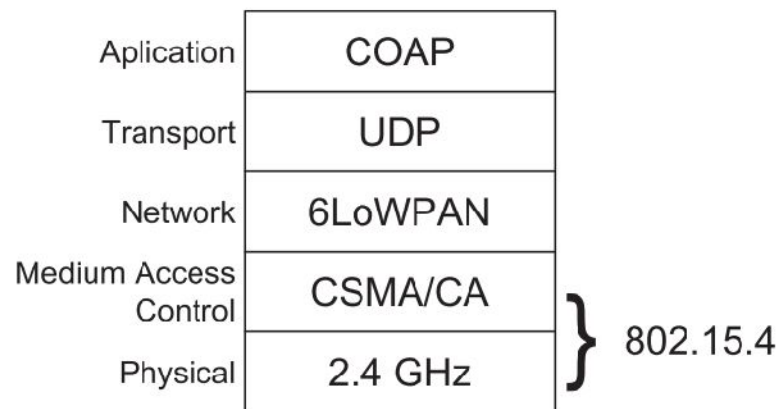


Figura 2 – Pilha de protocolos utilizada no trabalho de [Pacheco et al. \(2016\)](#)

Neste trabalho, propõe-se o uso do simulador Cooja para abordar a criação de ataques DoS/DDoS.

3 Descrição do ambiente Cooja/Contiki

Configurar grandes redes utilizando-se de dispositivos físicos pode representar um desafio. Assim, usar um simulador para desenvolver e testar sistemas pode ser bastante útil. Os simuladores, além de permitirem a rápida criação de protótipos e testes, também permitem uma execução de tarefas mais rápida quando comparados a ambientes reais (SEHGAL, 2013). Redes de sensores sem fio (do inglês *Wireless Sensor Networks* - WSN) tradicionalmente são limitadas a simular nós em um único nível de abstração, o que dificulta o desenvolvimento de sistemas visto que não é possível usar o mesmo simulador para algoritmos de alto e de baixo nível. No entanto, o simulador utilizado nesta monografia, que é fornecido juntamente com o sistema operacional (SO) Contiki, permite simulação com nós em níveis diferentes, tais como nível de rede, nível do sistema operacional e nível de código de máquina (ÖSTERLIND et al., 2006).

Neste capítulo, dados técnicos da arquitetura IoT - tais como os protocolos utilizados e sua finalidade - serão apresentados. Além disso, uma visão geral sobre o SO Contiki, que é um dos sistemas operacionais (SOs) criados para dispositivos IoT, será mostrada. Por último, o simulador Cooja, que será utilizado para replicar ataques DoS, será descrito.

3.1 Arquitetura IoT

Considerando que o TCP/IP é, atualmente, a pilha de protocolos mais utilizada para comunicação em rede, muitos acreditam que o IP pode ser o futuro da IoT. Os dispositivos IoT consistem, na maior parte, de nós compatíveis com o padrão IEEE 802.15.4, uma tecnologia de rede sem fio para aplicações de baixa potência e baixa taxa de transferência de dados, cuja cobertura é de apenas alguns metros. Neste padrão, a taxa de transferência máxima é de 250 kb/s, a potência máxima é de 1 mW e o tamanho máximo do pacote é de 127 bytes. Por conta destas características, protocolos baseados no IP, como o *Hypertext Transfer Protocol* (HTTP) não são adequados para este tipo de rede. A fim de enfrentar este problema, o *Internet Engineering Task Force* (IETF) assumiu a liderança na padronização de protocolos de comunicação para tais dispositivos e desenvolveu vários deles, incluindo o *Routing Protocol for Low Power and Lossy Networks* (RPL) e o CoAP (SHENG; YANG; YU, 2013) e (PÖTSCH; KULADINITHI; BECKER, 2012).

Desde o início da pesquisa do IETF em tecnologias relacionadas à IoT, o IPv6 foi selecionado como a única opção para habilitar sua comunicação sem fio. Suas principais características, como universalidade, extensibilidade e estabilidade, atraíram muita atenção e podem se tornar a solução ideal para o futuro da tecnologia da Internet. Para habilitar a conectividade IP em redes de sensores com recursos limitados, o grupo de

trabalho *IPv6 over Low Power Wireless Personal Area Networks* (6LoWPAN) da IETF cria e mantém as especificações que permitem o uso do IPv6 nas redes IEEE 802.15.4. O protocolo 6LoWPAN, especificamente, discute como aplicar o IPv6 às camadas MAC (do inglês *Medium Access Control* – Controle de Acesso ao Meio) e PHY (do inglês *Physical Layer* – Camada Física) dessas redes (SHENG; YANG; YU, 2013).

Existem dois desafios no uso do IPv6 em redes IEEE 802.15.4. Um deles está relacionado à fragmentação/desfragmentação de pacotes, visto que o tamanho máximo do pacote suportado por redes IEEE 802.15.4 é de 127 bytes e o valor mínimo da *maximum transmission unit* (MTU) especificada pelo IPv6 é de 1280 bytes. O outro é relacionado à compressão dos cabeçalhos IPv6 removendo o que não é necessário. Para resolver esses problemas, o 6LoWPAN cria uma camada de abstração entre o IPv6 e o IEEE 802.15.4 (SHENG; YANG; YU, 2013).

A fim de substituir o HTTP no ambiente IoT, o CoAP foi criado. Ele é um protocolo de camada de aplicação especialmente feito para satisfazer os requisitos de redes com limitação de recursos. Sendo um protocolo RESTful baseado no UDP, ele provê troca de mensagens assincronamente e de forma confiável, implementa baixo *overhead* nas mensagens, é simples e limita o uso de fragmentação. Por padrão, o CoAP suporta comunicação *peer-to-peer* mas pode ser expandido para suportar funções *one-to-many* via IP *multicast* (PÖTSCH; KULADINITHI; BECKER, 2012) e (CHEN; KUNZ, 2016).

O CoAP é baseado no mesmo modelo cliente/servidor do HTTP e possui um padrão de interação similar. Ele é principalmente usado para manipular recursos dos sensores (tais como umidade e temperatura) através dos métodos GET e PUT. Para fornecer um mecanismo confiável e leve na troca de mensagens, ele faz uso do tempo de espera exponencial e da retransmissão *stop-and-wait*. Além disso, é capaz de detectar mensagens duplicadas através do uso de identificadores únicos gerados aleatoriamente (PÖTSCH; KULADINITHI; BECKER, 2012).

RPL é um protocolo de roteamento e foi projetado para redes IPv6 de baixa potência com perda de pacotes (PÖTSCH; KULADINITHI; BECKER, 2012). Ele suporta três tipos de tráfego: *point-to-point* (entre sensores dentro da mesma rede), *point-to-multipoint* (de um ponto central de controle para um grupo de sensores na mesma rede) e *multipoint-to-point* (de um grupo de sensores da mesma rede para um ponto central de controle) (SHENG; YANG; YU, 2013).

Para calcular o caminho, o RPL utiliza um mecanismo de vetores de distância a partir da origem do roteamento no qual os nós formam um grafo acíclico e orientado ao destino (do inglês *destination-oriented acyclic graph* ou DODAG). O nó raiz (ou seja, o ponto central de controle) recebe as restrições do roteamento e filtra os nós que não as atendem, selecionando o caminho ótimo em relação às métricas recebidas. No estado estável, cada nó da rede já terá identificado um conjunto estável de nós pais e enviado

pacotes através do caminho escolhido para a raiz do DODAG (SHENG; YANG; YU, 2013). A Figura 3 exemplifica um DODAG.

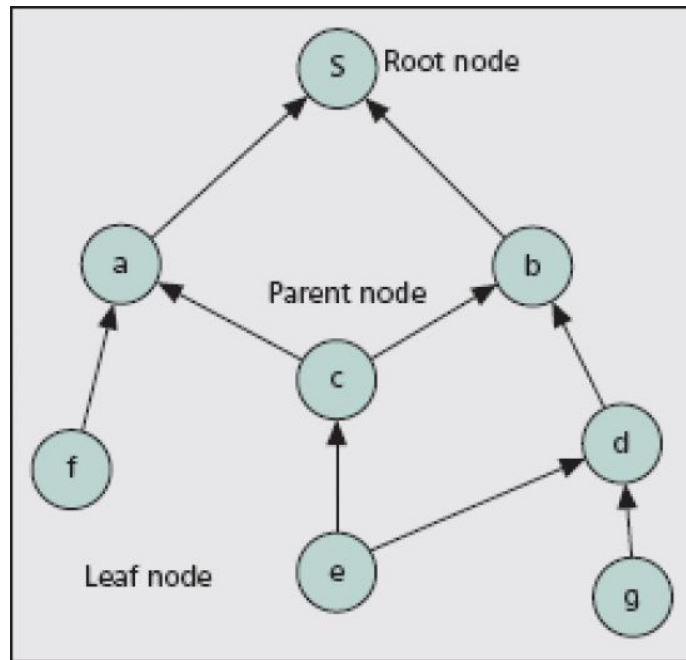


Figura 3 – Exemplo de DODAG. Retirada de Sheng, Yang e Yu (2013)

3.2 Contiki

Devido às limitações dos recursos físicos de dispositivos da IoT (descritas na sessão anterior), SOs específicos, que são menores e consomem menos recursos que os SOs comuns, devem ser utilizados. Os mais conhecidos são Contiki, TinyOS, Riot, Snappy e Raspbian (SANTOS et al., 2016). O Contiki, sendo um dos mais utilizados na comunidade acadêmica, é um SO desenvolvido usando a linguagem C, de código aberto, altamente eficiente tanto no uso de memória quanto no consumo de energia. Ele possui diversos mecanismos para alocação de memória, assim como diversos outros para estimativa de consumo de energia e para detecção dos locais onde o consumo ocorreu. Suas aplicações também são desenvolvidas em C, utilizam *threads* e são dirigidas a eventos. Ele conecta pequenos microcontroladores de baixo custo e baixa potência à Internet e além de suportar os protocolos IPv4, IPv6, TCP, UDP e HTTP, ele suporta também os protocolos de baixa potência mais recentes: 6LoWPAN, RPL e CoAP (CONTIKI, 2017).

No *Consumer Electronics Show* (CES) de 2014, alguns produtos comerciais que utilizam o Contiki como SO foram apresentados, entre eles o LIFX e o tado° (DUNKELS, 2017). LIFX são lâmpadas de LED conectadas à *Wi-Fi* que também se conectam ao *Apple Homekit*, *Google Assistant*, *Amazon Alexa*, *Nest*, *IFTT*, *Logitech Pop*, *Logitech Harmony* e *Samsung SmartThings*. Utilizando estes assistentes é possível controlar as lâmpadas

através de comandos de voz para ligar, desligar ou mudar a cor. Também é possível predefinir comandos para mudar o brilho e a temperatura da lâmpada. Além disso, é possível conectar as lâmpadas a outros sensores inteligentes, como o de movimento (LIFX, 2017).

O tado° conecta sistemas de aquecimento e resfriamento à internet. Ele regula automaticamente a temperatura de ares-condicionados e aquecedores baseado na previsão do tempo. Além disso, ele desliga estes sistemas quando a última pessoa sai da casa/empresa e religa quando detecta a primeira pessoa voltando ao ambiente (TADO, 2017).

3.3 Cooja

A fim de acelerar o desenvolvimento e os testes de aplicações que executam o Contiki, o simulador Cooja foi criado. Desenvolvido usando a linguagem Java, ele permite tanto a simulação de um único sensor quanto de uma rede de sensores. Periféricos, execução de aplicações e comunicação de rede são simulados em tempo real para quantos nós forem necessários. O usuário pode interagir com os nós usando uma interface gráfica (GUI) que permite o posicionamento dos nós, a modificação de suas entradas e a perturbação da comunicação da rede aumentando o nível de ruído, por exemplo (KUGLER; NORDHUS; ESKOFIER, 2013).

Como dito anteriormente, enquanto os simuladores tradicionais como NS-2, TOS-SIM e Avrora permitem simulações apenas em um nível de abstração (respectivamente nível de rede, nível do sistema operacional e nível de código de máquina), o Cooja permite simulação com nós em diferentes níveis. O nível ao qual a simulação é executada afeta tanto o nível ao qual o *software* pode ser desenvolvido quanto a eficiência de execução do simulador. Uma simulação ao nível de *hardware* permite a criação de *softwares* de baixo nível, porém o tempo de simulação e a complexidade da codificação são maiores, visto que apenas linguagens de baixo nível podem ser usadas. Da mesma forma, uma simulação em alto nível possui um tempo de simulação menor, porém apenas *softwares* de alto nível podem ser desenvolvidos. Assim, uma das vantagens de se utilizar o Cooja é a possibilidade de combinar diferentes níveis de abstração em uma única simulação. Em uma grande rede, por exemplo, alguns nós podem ser simulados ao nível de *hardware* enquanto os demais podem ser simulados ao nível da aplicação, o que combina as vantagens dos diferentes níveis (ÖSTERLIND et al., 2006). A Figura 4 apresenta os níveis de abstração dos simuladores aqui mencionados.

Para começar a utilizar o Cooja, a página do projeto Contiki (no endereço <http://www.contiki-os.org>) apresenta um tutorial com os passos necessários: inicia-se com o *download* do SO e da máquina virtual VMware e contempla a execução de uma simulação simples.

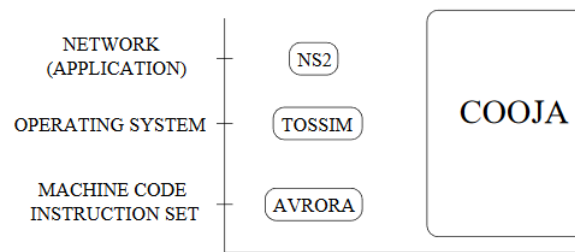


Figura 4 – Níveis de abstração do NS-2, TOSSIM, Avrora e Cooja. Retirada de [Österlind et al. \(2006\)](#)

Após baixar o Instant Contiki versão 2.6 e a máquina virtual VMware, basta iniciar o Contiki através da execução do InstantContiki2.6.vmx. A Figura 5 mostra a tela inicial do simulador, a Figura 6 mostra a tela inicial após criar a simulação e a Figura 7 mostra uma rede WSN com 5 motes (nós).



Figura 5 – Tela inicial do Cooja.

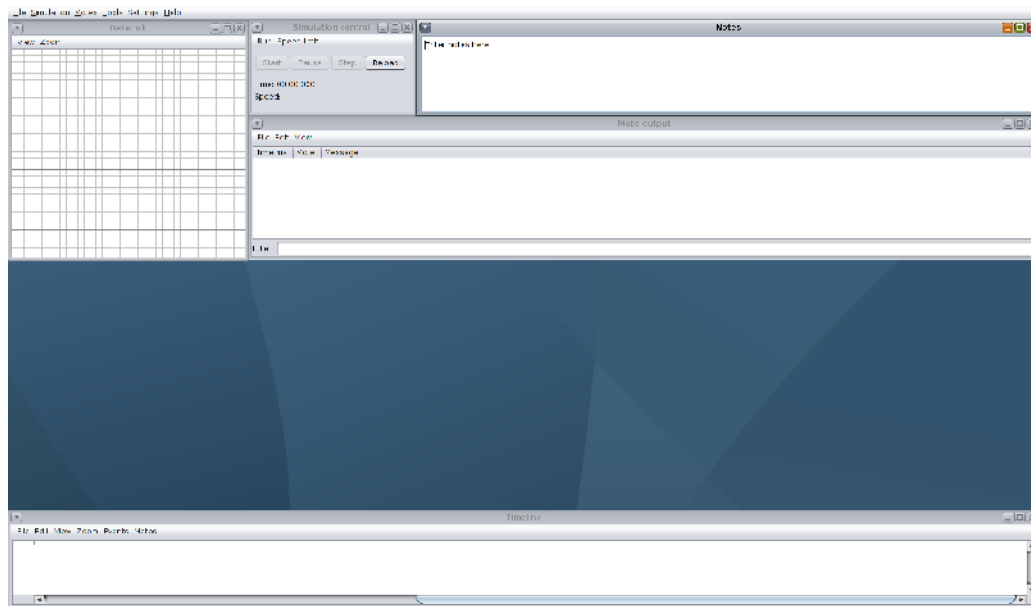


Figura 6 – Nova simulação.

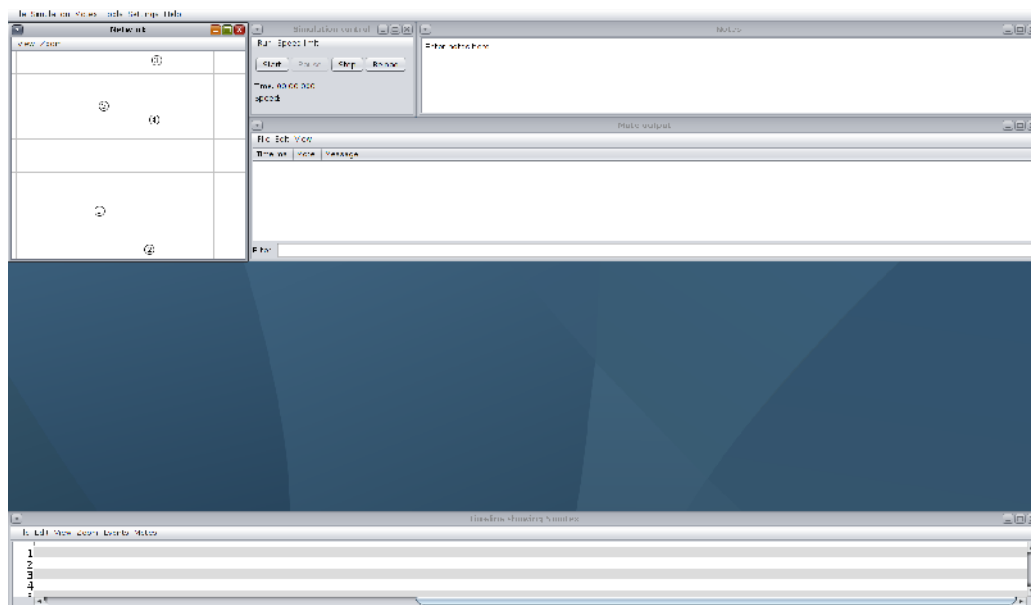


Figura 7 – Rede WSN com 5 nós.

4 Ataques DoS usando o Cooja

Nesta seção, os trabalhos de [Zhang e Green \(2015\)](#) e [Borgiani et al. \(2017\)](#) são apresentados. Ambos trazem um estudo sobre a segurança na IoT utilizando o Cooja – visto que é uma ferramenta criada para simular redes de dispositivos IoT – e influenciaram esta monografia. Sendo a segurança dos sensores IoT um fator crítico para seu uso e sendo os ataques DoS um dos problemas mais notórios relacionados à Internet das Coisas ([ZHANG; GREEN, 2015](#)), esses trabalhos propõem medidas validadas através de simulações no Cooja para evitar ou atenuar estes ataques.

No trabalho de [Zhang e Green \(2015\)](#), um algoritmo para prevenir e evitar ataques volumétricos de negação de serviço é proposto. A fim de validar este algoritmo, três cenários de rede foram criados no Cooja. Esses cenários são compostos por três tipos de nós: nó comum, nó legítimo e nó atacante.

1. Nó comum (*working node*): é responsável pela coleta de informações e pela execução de tarefas simples;
2. Nó legítimo (*legitimate node*): envia pacotes com pouca frequência (um pacote a cada 10 segundos) e com conteúdo coerente (o conteúdo dos pacotes difere entre si);
3. Nó atacante (*attacking node*): envia pacotes com alta frequência (1, 2 ou 3 segundos entre cada envio) e com o mesmo conteúdo em todos eles.

O primeiro cenário proposto possui 1 nó atacante e 1 nó comum em uma mesma rede IoT (Figura 8). Esse cenário procura examinar se o nó comum é capaz de identificar e rejeitar as requisições de serviço feitas pelo nó atacante. Os autores concluíram que, utilizando o algoritmo proposto, após atender à requisição do nó atacante pela primeira vez, o nó comum é capaz de identificar e rejeitar as próximas.

O segundo cenário possui 1 nó legítimo e 1 nó comum na mesma rede (Figura 9). A solicitação de serviços pelo nó legítimo começa após 10 segundos do início na simulação e neste caso não ocorre ataque. Os autores concluíram que, mesmo utilizando o algoritmo, o nó comum atende normalmente às requisições do nó legítimo.

A terceira simulação é composta por 1 nó legítimo, 1 nó comum e vários nós atacantes (Figura 10). Essa simulação exemplifica um ataque DDoS e seu objetivo é mostrar a distinção entre o tratamento das requisições de nós legítimos e o tratamento das requisições de nós atacantes pelo nó comum. O resultado do experimento mostrou aos autores que, enquanto as requisições feitas pelo nó legítimo são atendidas normalmente,

após a primeira requisição feita pelo nó atacante, o nó comum identifica e rejeita as próximas.

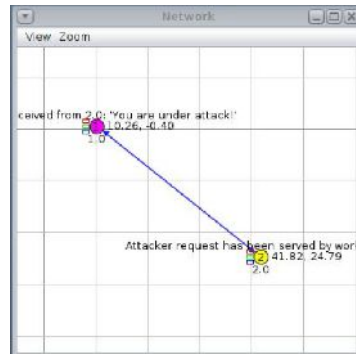


Figura 8 – Rede com um nó atacante e um nó comum. Retirada de Zhang e Green (2015).

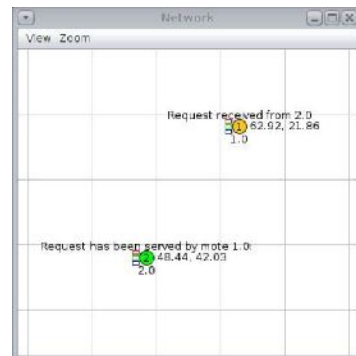


Figura 9 – Rede com um nó legítimo e um nó comum. Retirada de Zhang e Green (2015).

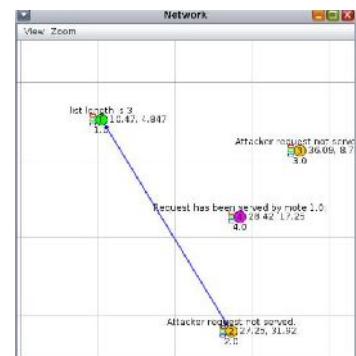


Figura 10 – Rede com um nó legítimo, um nó comum e múltiplos nós atacantes. Retirada de Zhang e Green (2015).

No trabalho de Borgiani et al. (2017), um mecanismo chamado ConCReCT (Controle de Congestionamento com Restrição do Ciclo de Trabalho) é apresentado. Este mecanismo tem como objetivo detectar ataques de negação de serviço volumétricos e atenuar seu efeito. Os ataques propostos nas simulações são baseados no envio de mensagens de *Hello* utilizando o protocolo UDP. As mensagens de *Hello* são mensagens com texto

simples e no caso do trabalho [Borgiani et al. \(2017\)](#) simulam as mensagens que deveriam ser enviadas pelos sensores de uma rede industrial para comunicar eventos como alterações de temperatura, pressão, presença de gás sulfídrico, entre outros. Essas mensagens são enviadas em intervalos de tempo pré-definidos, permitindo a previsão do tráfego na rede.

Quando um ataque é detectado, uma penalidade é aplicada ao nó atacante. O nó sendo atacado silencia o nó atacante por 300 segundos através do envio de uma mensagem de *wait*. Esse silenciamento afeta apenas a geração de novos pacotes pelo nó atacante, permitindo que ele continue a encaminhar pacotes provenientes de seus filhos. Assim, não há necessidade de reorganizar a topologia a fim de encontrar um novo pai para os filhos do nó malicioso, aumentando a tolerância a falhas da rede.

Durante a execução da simulação sem o ataque, as mensagens são enviadas a cada 1 minuto. Já durante o ataque, elas passam a ser enviadas a cada 1 segundo. As redes criadas são compostas de 1 nó RPL-UDP *server* e vários nós RPL-UDP *client*. Foram escolhidos 9 cenários diferentes para a simulação: redes com 10, 15 ou 20 nós e 1, 3 ou 5 nós simultaneamente atacando os demais. 20 simulações foram realizadas para cada um dos 9 cenários, cada uma com duas horas de duração. A lista dos cenários está na Figura 11 e a lista com os parâmetros utilizados nas simulações está na Figura 12.

Cenário	Nós da rede	Nós maliciosos
1	10	1
2	10	3
3	10	5
4	15	1
5	15	3
6	15	5
7	20	1
8	20	3
9	20	5

Figura 11 – Lista de cenários utilizados no trabalho de [Borgiani et al. \(2017\)](#).

Parâmetro	Valor
Área	100 x 100 m
Nós	10, 15 e 20
Topologia	Grade
Atacantes simultâneos	1, 3 e 5
Alcance de transmissão	45 m
Alcance de interferência	65 m
Atraso de inicialização de cada nó	1 milissegundo
Tempo de simulação	Até ocorrer a detecção
Frequência de <i>Hello</i> s	1/minuto
Frequência de <i>Hello</i> s durante ataque	1/segundo

Figura 12 – Lista de parâmetros utilizados nas simulações do trabalho de [Borgiani et al. \(2017\)](#).

Após analisar os resultados das simulações, os autores concluíram que o Con-CReCT tem uma taxa de detecção de 100% em ambientes restritos a 20 nós, com até 5

```

else if (ev == sensors_event && data == &button_sensor) {
    PRINTF("SISTEMA: Iniciando Ataque\n");
    normalInterval=attackInterval;
    etimer_set(&periodic, normalInterval * CLOCK_SECOND);
}

```

Figura 13 – Código utilizado para simular o ataque de negação de serviço.

atacantes simultâneos. Essa taxa, no entanto, pode diminuir em função do atraso ou da perda de mensagens enviadas pelos nós ao servidor ou de mensagens de *wait* enviadas aos atacantes a fim de silenciá-los.

4.1 Experimentos

Na seção anterior foram apresentados os trabalhos de [Zhang e Green \(2015\)](#) e [Borgiani et al. \(2017\)](#). Ambos simulam ataques volumétricos de negação de serviço e propõem, respectivamente, um algoritmo para prevenir e evitar tais ataques e outro algoritmo para detectar e atenuar seus efeitos. Os dois trabalhos validam os algoritmos propostos, mas não apresentam a análise da performance dos sensores sendo atacados durante o ataque. Assim, esta monografia propõe a simulação de ataques de negação de serviço volumétricos – assim como os trabalhos da seção anterior – e acrescenta a análise do desempenho dos sensores sendo atacados antes e durante o ataque para verificar, de fato, o potencial da IoT para ataques de negação de serviço.

As simulações aqui propostas possuem 2 tipos de nós: nó RPL-UDP *sink* e nó RPL-UDP *sender*. O nó *sink* é o alvo do ataque enquanto os nós *sender* podem ser tanto os atacantes quanto outros nós presentes na rede. O ataque é baseado no envio de pacotes *Hello* pelos nós *sender* ao nó *sink* e tem o intuito de comprometer a coleta de informações do nó sendo atacado.

Assim como no trabalho de [Borgiani et al. \(2017\)](#), o ataque aqui é causado pelo aumento na frequência do envio de pacotes *Hello*. Enquanto o RPL-UDP *client* nativo do Contiki envia as mensagens a uma taxa de uma mensagem por minuto, sua versão modificada – e utilizada para criar o ataque – envia mensagens a uma taxa de uma mensagem por segundo.

O ataque ocorre quando o usuário clica no botão do *sender* atacante. O código utilizado para simular o ataque é apresentado na Figura 13. Nele, ao encontrar um evento de clique, a taxa de envio de pacotes *Hello* passa de um envio por minuto para um envio por segundo. Todo o código desenvolvido com a finalidade de criar o ataque está disponível no apêndice A.

Os experimentos são compostos por 5 cenários. No primeiro cenário, uma rede (Figura 14) composta de 1 nó *sink* (alvo) e um nó *sender* (atacante) é criada. No segundo

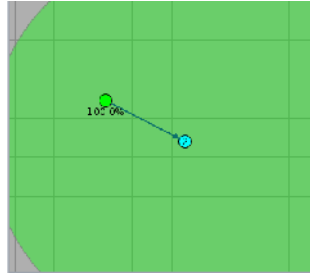


Figura 14 – Rede composta de 1 alvo (nó 2) e 1 atacante (nó 1).

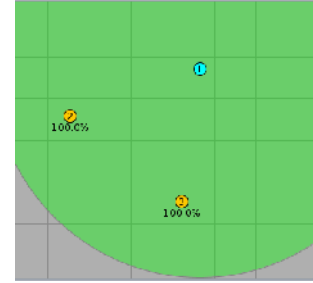


Figura 15 – Rede composta de 1 alvo (nó 1), 1 atacante (nó 2) e 1 outro nó.

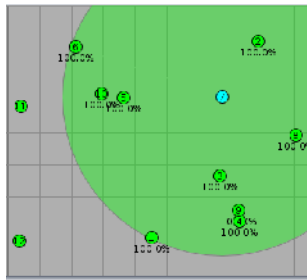


Figura 16 – Rede composta de 1 alvo (nó 7), 1 atacante (nó 2) e 10 outros nós.

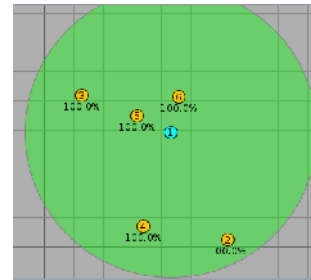


Figura 17 – Rede composta de 1 alvo (nó 1) e 5 atacantes (nós 2, 3, 4, 5 e 6).

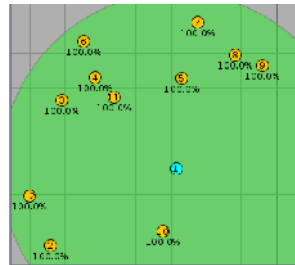


Figura 18 – Rede composta de 1 alvo (nó 1), 5 atacantes (nós 2, 3, 4, 5 e 6) e 6 outros nós.

cenário, a rede (Figura 15) proposta possui 1 nó *sink* (alvo) e 2 nós *sender*, sendo 1 destes um atacante. No terceiro, a rede (Figura 16) é formada por 1 nó *sink* (alvo) e 11 nós *sender*, sendo 1 deles um atacante. Para o quarto caso, 1 nó *sink* (alvo) e 5 nós *sender* (atacantes) foram utilizados (Figura 17). O quinto e último cenário (Figura 18) possui 1 nó *sink* (alvo) e 11 nós *sender*, sendo 5 deles atacantes. A Tabela 1 apresenta uma síntese destes cenários.

O tempo de simulação para cada cenário varia de acordo com o número de nós. Para cada nó presente na rede, 10 minutos são adicionados ao tempo de execução. Destes 10 minutos, 5 são utilizados na primeira fase da simulação - onde o código RPL-UDP nativo do Contiki é utilizado, e 5 na segunda fase - onde o código RPL-UDP modificado

Tabela 1 – Cenários utilizados nas simulações

Cenário	Nós
Cenário 1	1 nó alvo e 1 nó atacante
Cenário 2	1 nó alvo, 1 nó atacante e 1 nó comum
Cenário 3	1 nó alvo, 1 nó atacante e 10 nós comuns
Cenário 4	1 nó alvo e 5 nós atacantes
Cenário 5	1 nó alvo, 5 nós atacantes e 6 nós comuns

Figura 19 – Dispositivo real emulado pelo *tmote sky*. Retirado de [IV \(2017\)](#).

e habilitado a criar ataques DoS, é utilizado. Por exemplo, o cenário 1 possui dois nós, assim, 10 minutos serão gastos na simulação antes do ataque (5 minutos para cada nó) e mais 10 minutos durante o ataque (5 minutos para cada nó), totalizando 20 minutos. A Tabela 2 mostra o tempo gasto na simulação de cada cenário.

Tabela 2 – Tempo gasto para simular cada cenário antes e durante o ataque e tempo total da simulação para cada cenário

Cenário	Tempo antes do ataque	Tempo durante o ataque	Tempo total
Cenário 1 (2 nós)	10min	10min	20min
Cenário 2 (3 nós)	15min	15min	30min
Cenário 3 (12 nós)	1hr	1hr	2hrs
Cenário 4 (6 nós)	30min	30min	1hr
Cenário 5 (12 nós)	1hr	1hr	2hrs

Todas as simulações foram criadas utilizando o sensor *tmote sky* (visto na Figura 19), de propriedade da *Mote IV Corporation*, de potência ultra-baixa, compatível com o padrão 802.15.4, baseado no difundido microcontrolador MSP430 e com recursos de memória RAM (10k) e armazenamento flash (48k) ([BORGIANI et al., 2017](#)). Além disso, o *tmote sky* possui dois sensores de luz fotodiodos - dispositivos que convertem luz em corrente elétrica. Um deles detecta radiação e o outro detecta o espectro visível, incluindo infravermelho. Ele possui também um sensor de umidade/temperatura que é opcional e deve ser calibrado para produzir uma saída digital. Assim, esse dispositivo é capaz de mensurar umidade relativa, temperatura e luz ([IV, 2017](#)), o que permite, por exemplo, o monitoramento de informações críticas durante um incêndio, tais como fumaça, luz e fogo ([MOORE, 2017](#)).

Para cada cenário, três simulações foram criadas. Os parâmetros gerais utilizados

em cada uma delas são apresentados na Tabela 3.

Tabela 3 – Parâmetros utilizados nas simulações

Parâmetro	Valor
Topologia	Grade
Número de simulações para cada cenário	3
Tempo de simulação total	20min, 30min, 2hrs, 1hr, 2hrs
Frequência de <i>Hello</i> s antes do ataque	1/minuto
Frequência de <i>Hello</i> s durante o ataque	1/segundo

A Seção 4.2 discute os resultados sob o ponto de vista de uma das três simulações que foram criadas para cada cenário. Os resultados das outras duas execuções são mostrados no Apêndice B.

4.2 Resultados

Para avaliar o impacto do possível ataque, informações dos sensores presentes no nó alvo foram capturadas em um contexto sem ataque e em outro com ataque. Os gráficos 20 ao 34 mostram os dados coletados pelos sensores de *battery voltage*, *light 1* e *light 2* para cada cenário, antes do ataque e durante ele. A coleta de dados é feita através do envio de pacotes do nó *sink* para a rede. Esse envio ocorre a cada 60 segundos e portanto a coleta de dados também ocorre a cada 60 segundos.

Ao observar os gráficos com os dados coletados da *battery voltage* (Figura 20), *light 1* (Figura 21) e *light 2* (Figura 22) para o cenário 1, é possível visualizar que a frequência da coleta é alta antes do ataque e baixa durante ele. Uma possível explicação para esse comportamento considera que antes do ataque a frequência de envio de pacotes *Hello* era de 1/minuto e durante o ataque passou a ser de 60/minuto, aumentando em 60 vezes o envio desses pacotes ao nó alvo. Assim, o aumento na frequência de envio desses pacotes é diretamente proporcional à diminuição da frequência de coleta de informações pelos sensores do nó alvo.

Ao observar os gráficos com os dados coletados da *battery voltage* (Figura 23), *light 1* (Figura 24) e *light 2* (Figura 25) para o cenário 2, é possível visualizar que a frequência da coleta é alta antes do ataque e baixa durante ele. Uma possível explicação para esse comportamento considera que antes do ataque a frequência de envio de pacotes *Hello* era de 2/minuto e durante o ataque passou a ser de 61/minuto, indicando um aumento de aproximadamente 30 vezes no seu envio ao nó alvo. Assim, o aumento na frequência de envio desses pacotes é diretamente proporcional à diminuição da frequência de coleta de informações pelos sensores do nó alvo.

Ao observar os gráficos com os dados coletados da *battery voltage* (Figura 26), *light 1* (Figura 27) e *light 2* (Figura 28) para o cenário 3, é possível visualizar que, apesar da

frequência da coleta nos dois casos ser próxima, ainda há uma ligeira redução durante o ataque. Para explicar esse comportamento, basta considerar que antes do ataque a frequência de envio de pacotes *Hello* era de 11/minuto e durante o ataque passou a ser de 70/minuto. Isso mostra um aumento de aproximadamente 7 vezes no envio desses pacotes ao nó alvo. Neste cenário, o aumento na frequência de envio de pacotes é menor em relação aos dois cenários anteriores e da mesma forma, a coleta dos dados durante o ataque ocorre com uma frequência mais próxima à coleta antes do ataque. Assim, o aumento na frequência de envio desses pacotes é diretamente proporcional à diminuição da frequência de coleta de informações pelos sensores do nó alvo.

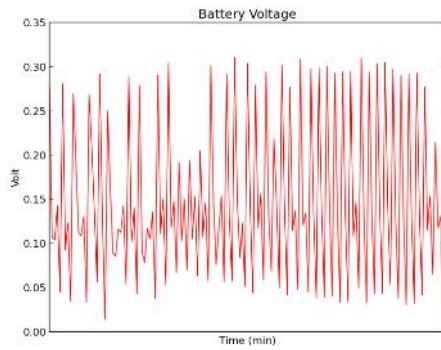
Ao observar os gráficos com os dados coletados da *battery voltage* (Figura 29), *light 1* (Figura 30) e *light 2* (Figura 31) para o cenário 4, é possível visualizar que a frequência da coleta é alta antes do ataque e baixa durante ele. Uma possível explicação para esse comportamento considera que antes do ataque a frequência de envio de pacotes *Hello* era de 5/minuto e durante o ataque passou a ser de 300/minuto, um aumento de aproximadamente 60 vezes no envio desses pacotes ao nó alvo. Assim, o aumento na frequência de envio desses pacotes é diretamente proporcional à diminuição da frequência de coleta de informações pelos sensores do nó alvo.

Ao observar os gráficos com os dados coletados da *battery voltage* (Figura 32), *light 1* (Figura 33) e *light 2* (Figura 34) para o cenário 5, é possível visualizar que a frequência da coleta é alta antes do ataque e baixa durante ele. Uma possível explicação para esse comportamento considera que antes do ataque a frequência de envio de pacotes *Hello* era de 11/minuto e durante o ataque passou a ser de 306/minuto. Isso mostra um aumento de aproximadamente 30 vezes no envio desses pacotes ao nó alvo. Assim, o aumento na frequência de envio desses pacotes é diretamente proporcional à diminuição da frequência de coleta de informações pelos sensores do nó alvo.

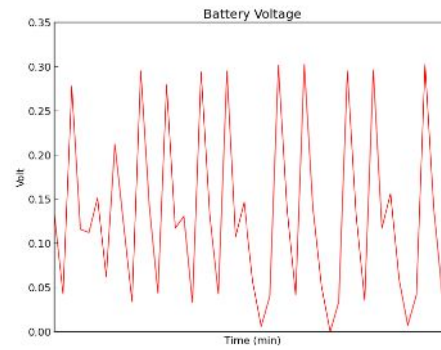
A Tabela 4 sintetiza as informações da frequência de envio de pacotes *Hello* para cada cenário. Nela são apresentadas as frequência antes e durante o ataque. Além disso, ela também exibe o aumento dessa frequência no período com ataque em relação ao período sem ataque.

Como pode ser visto na Tabela 4, para os cenários 1 e 4 há um aumento de 60 vezes no envio de pacotes *Hello* durante o ataque. Para os cenários 2 e 5 o aumento é de 30 vezes e para o cenário 3 o aumento é de apenas 7 vezes. Em condições normais da rede, os sensores de *battery voltage*, *light 1* e *light 2* poderiam ser acessados normalmente. Já com a rede estressada, o acesso a eles aconteceu em uma frequência bem menor, o que indica uma sobrecarga nesse nó e, portanto, um ataque de negação de serviço.

Os experimentos aqui conduzidos foram simulados utilizando o Cooja, que é um simulador criado para simular redes IoT. Apesar de ser possível reproduzir redes IoT e simular ataques de negação de serviço no Cooja facilmente, o funcionamento e a coleta

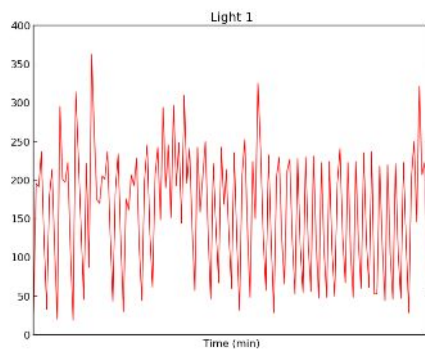


(a) Battery Voltage - antes do ataque

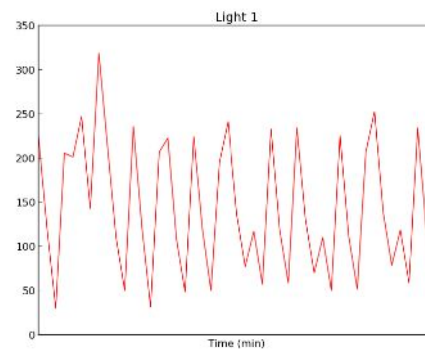


(b) Battery Voltage - durante o ataque

Figura 20 – Cenário 1: Battery Voltage antes e durante o ataque

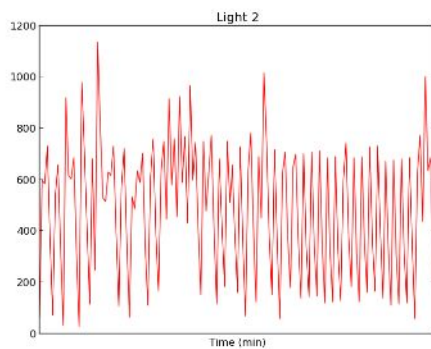


(a) Light 1 - antes do ataque

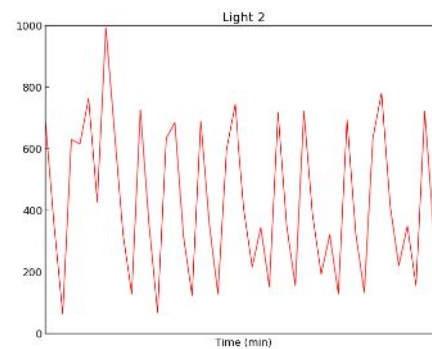


(b) Light 1 - durante o ataque

Figura 21 – Cenário 1: Light 1 antes e durante o ataque

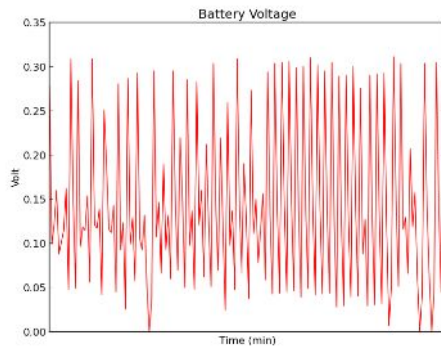


(a) Light 2 - antes do ataque

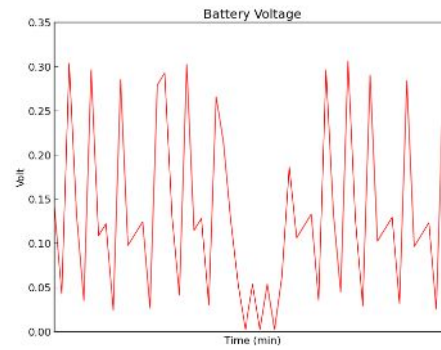


(b) Light 2 - durante o ataque

Figura 22 – Cenário 1: Light 2 antes e durante o ataque

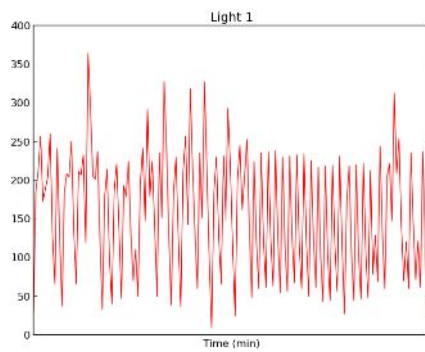


(a) Battery Voltage - antes do ataque

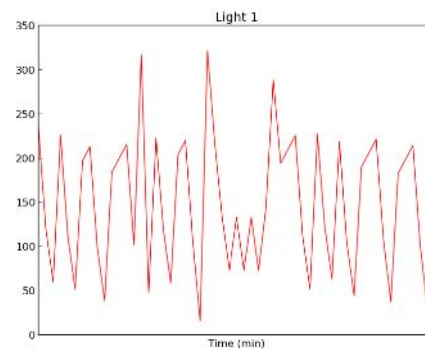


(b) Battery Voltage - durante o ataque

Figura 23 – Cenário 2: Battery Voltage antes e durante o ataque

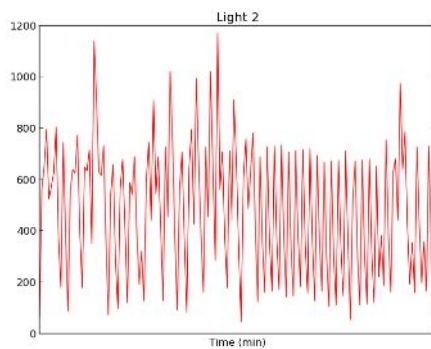


(a) Light 1 - antes do ataque

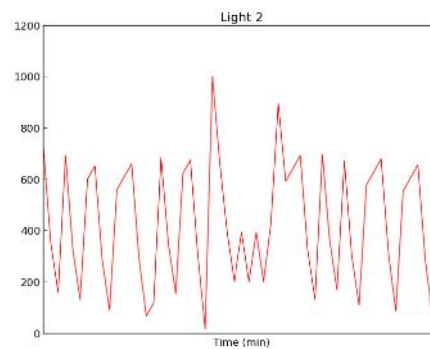


(b) Light 1 - durante o ataque

Figura 24 – Cenário 2: Light 1 antes e durante o ataque

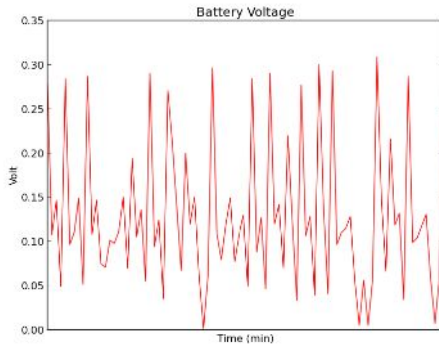


(a) Light 2 - antes do ataque

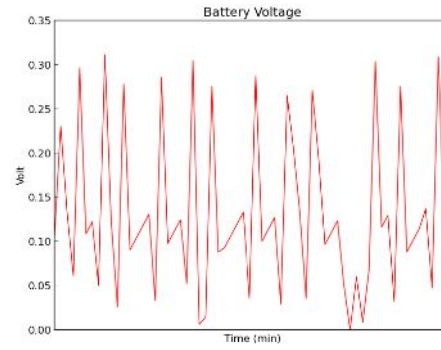


(b) Light 2 - durante o ataque

Figura 25 – Cenário 2: Light 2 antes e durante o ataque

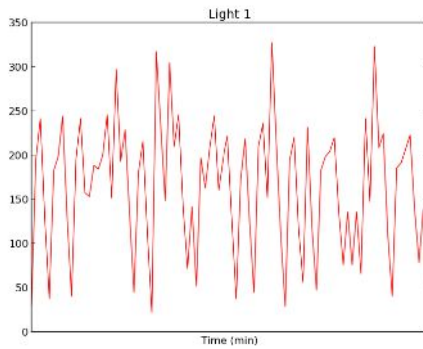


(a) Battery Voltage - antes do ataque

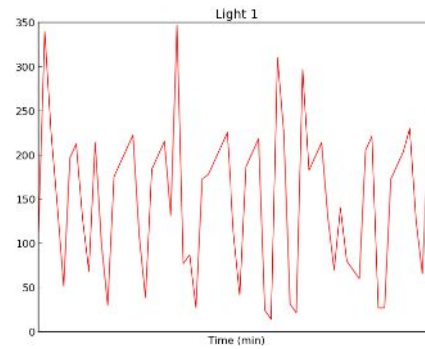


(b) Battery Voltage - durante o ataque

Figura 26 – Cenário 3: Battery Voltage antes e durante o ataque

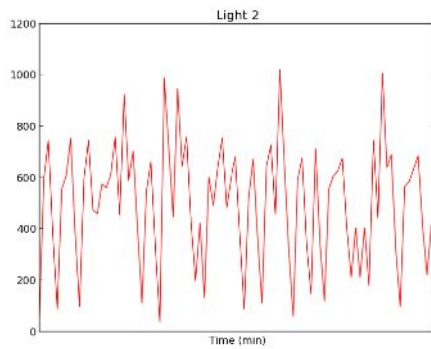


(a) Light 1 - antes do ataque

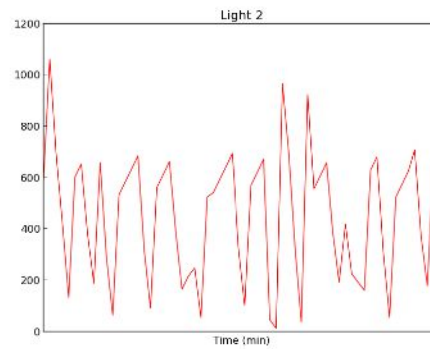


(b) Light 1 - durante o ataque

Figura 27 – Cenário 3: Light 1 antes e durante o ataque

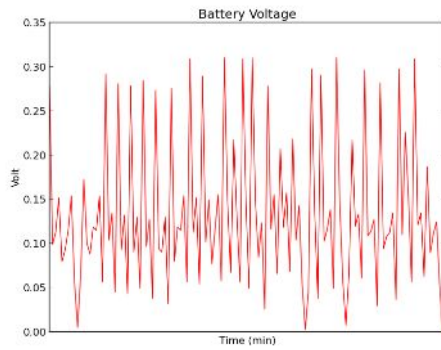


(a) Light 2 - antes do ataque

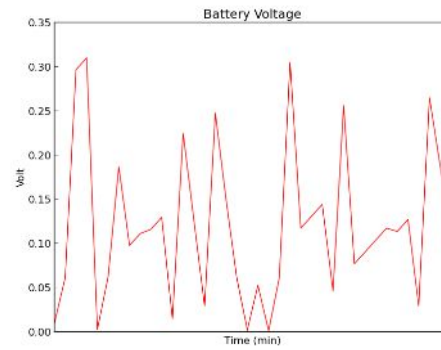


(b) Light 2 - durante o ataque

Figura 28 – Cenário 3: Light 2 antes e durante o ataque

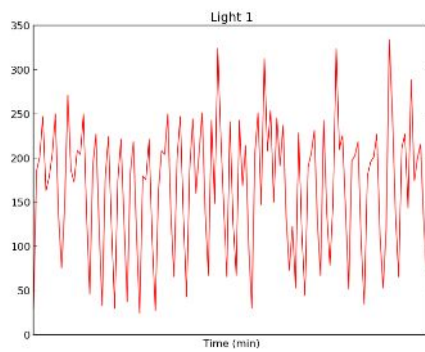


(a) Battery Voltage - antes do ataque

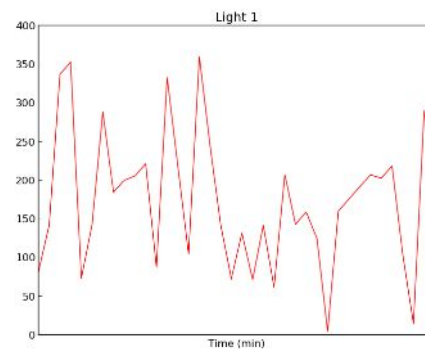


(b) Battery Voltage - durante o ataque

Figura 29 – Cenário 4: Battery Voltage antes e durante o ataque

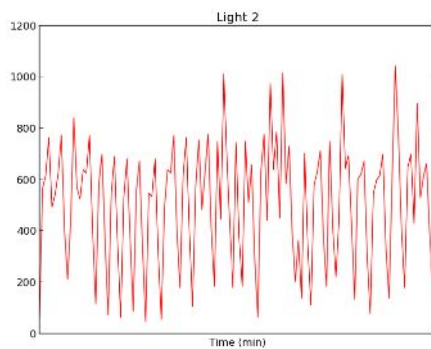


(a) Light 1 - antes do ataque

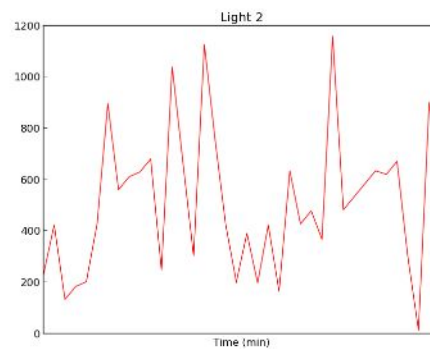


(b) Light 1 - durante o ataque

Figura 30 – Cenário 4: Light 1 antes e durante o ataque

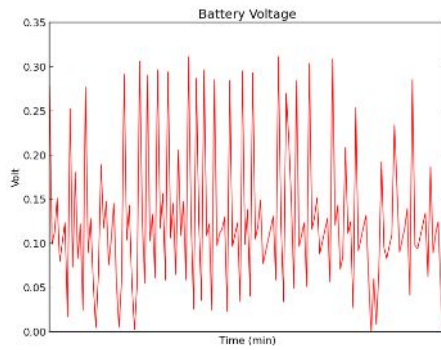


(a) Light 2 - antes do ataque

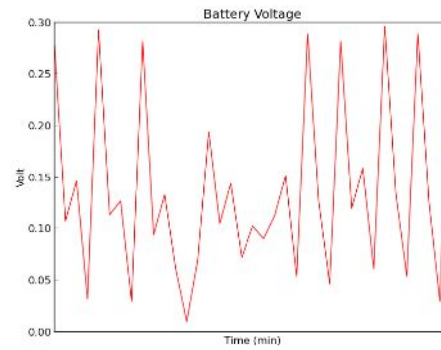


(b) Light 2 - durante o ataque

Figura 31 – Cenário 4: Light 2 antes e durante o ataque

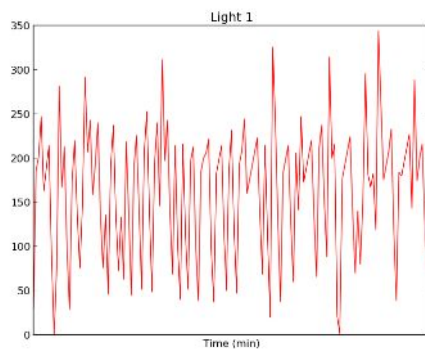


(a) Battery Voltage - antes do ataque

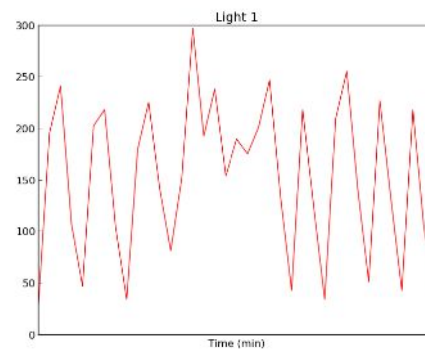


(b) Battery Voltage - durante o ataque

Figura 32 – Cenário 5: Battery Voltage antes e durante o ataque

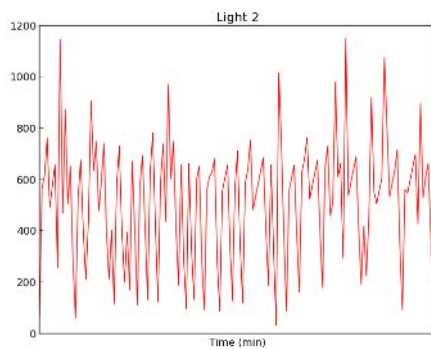


(a) Light 1 - antes do ataque

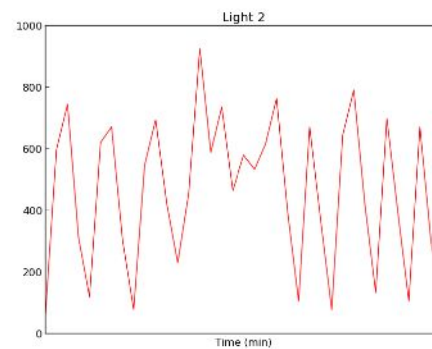


(b) Light 1 - durante o ataque

Figura 33 – Cenário 5: Light 1 antes e durante o ataque



(a) Light 2 - antes do ataque



(b) Light 2 - durante o ataque

Figura 34 – Cenário 5: Light 2 antes e durante o ataque

Cenário	frequência antes do ataque	frequência durante o ataque	aumento
1	1/minuto	60/minuto	60 vezes
2	2/minuto	61/minuto	30 vezes
3	11/minuto	70/minuto	7 vezes
4	5/minuto	300/minuto	60 vezes
5	11/minuto	306/minuto	30 vezes

Tabela 4 – Frequência de envio de pacotes *Hello*.

de dados dos nós simulados não é algo intuitivo. Além disso, a documentação disponível abrange apenas exemplos e casos de uso simples, deixando os casos mais complexos sob responsabilidade das listas de e-mail do projeto. Entretanto, ele é um simulador cujas redes e seus nós são baseados na arquitetura IoT, o que evita um trabalho extensivo na configuração desse tipo de ambiente em outros simuladores. Além disso, ele permite simulações com nós em diferentes níveis de abstração, ao contrário de outros simuladores tradicionais.

Como foi observado através dos resultados desses experimentos, existem vulnerabilidades na segurança da IoT. O *tmote sky* foi o dispositivo avaliado neste trabalho e os experimentos aqui conduzidos mostraram que, além de ser possível iniciar ataques a partir dele, ataques recebidos por ele prejudicam visivelmente seu funcionamento. Isso fica claro ao analisar a frequência da coleta de dados pelos sensores do dispositivo em um momento sem ataque e em outro com ataque, visto que essa frequência diminui na mesma proporção em que o envio de pacotes *Hello* aumenta (esse aumento caracteriza o ataque aqui estudado: ataque volumétrico).

Neste trabalho, os ataques volumétricos foram gerados por até 5 nós maliciosos atacando o nó alvo simultaneamente. Mesmo com esse número baixo de atacantes foi possível verificar a viabilidade do ataque, visto que o funcionamento do nó alvo foi afetado. Assim, ao considerar um ataque de maior proporção, como o ocorrido em 2016 que utilizou o *malware* Mirai, poderia ser possível inviabilizar completamente o uso do alvo do ataque.

A fim de prevenir/mitigar esses ataques, [Nakamura e Geus \(2010\)](#) sugerem o uso de *firewall*, anti-vírus, políticas de segurança e sistemas de detecção de intrusão. Especificamente, os algoritmos propostos por [Borgiani et al. \(2017\)](#) e [Zhang e Green \(2015\)](#) são capazes de prevenir, atenuar e detectar tais ataques.

5 Conclusão

O objetivo deste trabalho foi verificar o potencial da IoT para ataques de negação de serviço. No capítulo 2, uma revisão da literatura foi apresentada. Informações sobre IoT, segurança da informação e ataques de negação de serviço foram compiladas.

O capítulo 3 apresentou uma descrição sobre a arquitetura de rede da IoT, em especial sobre os protocolos utilizados: o 6LoWPAN - que discute como aplicar o IPv6 às camadas MAC e PHY dessas redes, o CoAP - que substitui o HTTP e provê troca de mensagens assincronamente e de forma confiável, e o RPL - que é um protocolo de roteamento e foi projetado para redes IPv6 de baixa potência com perda de pacotes. Apresentou também as ferramentas utilizadas neste projeto: o Sistema Operacional Contiki - criado especialmente para dispositivos com limitação de recursos, e o simulador Cooja - utilizado para simulação de redes compostas por dispositivos baseados no SO Contiki.

O capítulo 4 reuniu dois trabalhos encontrados na literatura onde ataques de negação de serviço volumétricos foram simulados utilizando o Cooja. No trabalho de [Zhang e Green \(2015\)](#), um algoritmo para prevenir e evitar tais ataques foi proposto e validado. No trabalho de [Borgiani et al. \(2017\)](#), um algoritmo para detectar e atenuar seus efeitos foi sugerido e também validado. Além disso, esse capítulo apresentou também o desenvolvimento deste trabalho onde, assim como os trabalhos de [Zhang e Green \(2015\)](#) e [Borgiani et al. \(2017\)](#), ataques de negação de serviço volumétricos foram simulados utilizando o Cooja. Entretanto, ao contrário dos trabalhos apresentados, esta monografia fez uma análise do desempenho dos nós alvo antes e durante o ataque, mostrando por meio de gráficos a queda na frequência de coleta de dados do nó alvo entre o primeiro e o segundo momento.

Como trabalhos futuros, sugere-se a medição de outros parâmetros do nó alvo, como latência, temperatura, umidade, uso de processamento, uso de memória, etc. Sugere-se também verificar o comportamento do nó alvo perante diferentes tipos de ataques de negação de serviço. Ainda, sugere-se simular tais ataques partindo de sensores presentes em uma rede IoT e tendo como alvo máquinas com poder computacional maior, tais como computadores pessoais ou servidores.

Referências

- ALABA, F. A. et al. Internet of things security: A survey. *Journal of Network and Computer Applications*, v. 88, p. 10–28, 2017. Citado 2 vezes nas páginas 9 e 15.
- ANGRISHI, K. Turning internet of things(iot) into internet of vulnerabilities (iov) : Iot botnets. p. 1–17, 2017. Citado 4 vezes nas páginas 9, 13, 14 e 15.
- BERTINO, E. Botnets and internet. *IEEE Computer Society*, p. 76–79, 2017. Citado na página 13.
- BISHOP, M. *Computer Security: Art and Science*. [S.l.]: Addison Wesley, 2002. Citado na página 11.
- BORGIANI, V. et al. Concret: Um mecanismo para mitigação dos efeitos de ataques de negação de serviço em internet das coisas industriais. *XVII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais - SBSeg 2017*, p. 84–97, 2017. Citado 8 vezes nas páginas 4, 26, 27, 28, 29, 31, 39 e 40.
- CHEN, Y.; KUNZ, T. Performance evaluation of iot protocols under a constrained wireless access network. *Selected Topics in Mobile & Wireless Networking (MoWNeT), 2016 International Conference*, 2016. Citado na página 21.
- CONTIKI. *Contiki*. [S.l.], 2017. Disponível em: <<http://www.contiki-os.org/index.html>>. Acesso em: 26 out. 2017. Citado na página 22.
- CUSACK, B.; TIAN, Z.; KYAW, A. *Identifying DOS and DDOS attack origin: IP traceback methods comparison and evaluation for IoT*. Cham: Springer, Cham, 2017. Citado na página 12.
- DLAMINI, M.; ELOFF, J.; ELOFF, M. Information security: The moving target. *Computers & Security*, v. 28, n. 3-4, p. 189–198, 2009. Citado na página 11.
- DUNKELS, A. *Contiki-based Products at CES 2014*. [S.l.], 2017. Disponível em: <<http://contiki-os.blogspot.com.br/>>. Acesso em: 18 dec. 2017. Citado na página 22.
- FEITOSA, E.; SOUTO, E.; SADOK, D. Tráfego internet não desejado: Conceitos, caracterização e soluções. *Proc. VIII SBSeg, SBC*, p. 91–137, 2008. Citado na página 12.
- GREENBERG, A. Hackers remotely kill a jeep on the highway—with me in it. *Wired*, v. 7, p. 21, 2015. Citado na página 16.
- HENGST, K. Ddos through the internet of things. p. 1–8, 2016. Citado 4 vezes nas páginas 4, 9, 17 e 18.
- IV, M. *tmote sky datasheet*. [S.l.], 2017. Disponível em: <<http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf>>. Acesso em: 18 dec. 2017. Citado 2 vezes nas páginas 4 e 31.
- KNAPP, K. J. et al. Information security: management’s effect on culture and policy. *Information Management & Computer Security*, v. 14, n. 1, p. 24–36, 2006. Citado na página 11.

- KRITZINGER, E.; SMITH, E. Information security management: An information security retrieval and awareness model for industry. *Computers & Security*, v. 27, n. 5-6, p. 224–231, 2008. Citado na página 11.
- KUGLER, P.; NORDHUS, P.; ESKOFIER, B. Shimmer, cooja and contiki: A new toolset for the simulation of on-node signal processing algorithms. *2013 IEEE International Conference on Body Sensor Networks, BSN 2013*, 2013. Citado na página 23.
- LAUFER, R. P. et al. Negação de serviço: Ataques e contramedidas. *Livro Texto dos Mini-cursos do V Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, p. 359–421, 2005. Citado 2 vezes nas páginas 12 e 14.
- LIFX. *LIFX*. [S.l.], 2017. Disponível em: <<https://lifax.com/>>. Acesso em: 18 dec. 2017. Citado na página 23.
- LINDQVIST, U.; NEUMANN, P. G. The future of the internet of things. *Communications of the ACM*, v. 60, n. 2, p. 26–30, 2017. Citado 3 vezes nas páginas 15, 16 e 18.
- MANSFIELD-DEVINE, S. Ddos goes mainstream: how headline-grabbing attacks could make this threat an organisation's biggest nightmare. *Network Security*, v. 2016, n. 11, p. 7–13, 2016. Citado 2 vezes nas páginas 10 e 14.
- MIANI, R. S. *Um Estudo sobre Métricas e Quantificação em Segurança da Informação*. Tese (Doutorado) — Faculdade de Engenharia Elétrica e de Computação, Campinas, SP, 2013. Citado na página 11.
- MOORE, S. *Tmote sky*. [S.l.], 2017. Disponível em: <<https://wirelessensornetworks.weebly.com/blog/tmote-sky>>. Acesso em: 18 dec. 2017. Citado na página 31.
- NAKAMURA, E. T.; GEUS, P. L. d. *Segurança de Redes em Ambientes Cooperativos*. São Paulo, SP: Novatec Ediwra Ltda., 2010. Citado 2 vezes nas páginas 12 e 39.
- ÖSTERLIND, F. et al. Demo abstract: Cross-level simulation in cooja. *Proceedings of the first IEEE International Workshop on Practical Issues in Building Sensor Network Applications*, 2006. Citado 4 vezes nas páginas 4, 20, 23 e 24.
- PACHECO, L. A. B. et al. Evaluation of distributed denial of service threat in the internet of things. *15th International Symposium on Network Computing and Applications Evaluation*, p. 89–92, 2016. Citado 5 vezes nas páginas 4, 9, 15, 18 e 19.
- PEREIRA, H. *Sistema de Detecção de Intrusão para Serviços WEB Baseado em Anomalias*. Dissertação (Dissertação de Mestrado) — Pontifícia Universidade Católica do Paraná, 2011. Citado na página 13.
- PÖTSCH, T.; KULADINITHI, K.; BECKER, M. Performance evaluation of coap using rpl and lpl in tinyos. *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference*, 2012. Citado 2 vezes nas páginas 20 e 21.
- RAYES, A.; SALAM, S. *Internet of Things - From Hype to Reality*. Cham, Switzerland: Springer International Publishing, 2017. Acesso em: 2017-05-09. Citado na página 9.
- SANTOS, B. P. et al. Internet das coisas: da teoria à prática. *Livro de MinicursosSBRC 2016*, p. 1–50, 2016. Citado na página 22.

- SEHGAL, A. Using the contiki cooja simulator. 2013. Citado na página 20.
- SHENG, Z.; YANG, S.; YU, Y. A survey on the ietf protocol suite for the internet of things: standards, challenges, and opportunities. *IEEE Wireless Communications*, v. 20, n. 6, p. 91–98, 2013. Citado 4 vezes nas páginas 4, 20, 21 e 22.
- 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 9.
- TADO. *tado°*. [S.l.], 2017. Disponível em: <<https://www.tado.com>>. Acesso em: 18 dec. 2017. Citado na página 23.
- TRAORE, I.; AWAD, A.; WOUNGANG, I. *Information Security Practices*. Cham: Springer International Publishing, 2017. Acesso em: 09 maio 2017. Citado na página 11.
- XIA, F. et al. Internet of things. *INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS Int. J. Commun. Syst. Int. J. Commun. Syst*, v. 25, n. 25, p. 1101–1102, 2012. Citado na página 14.
- ZHANG, C.; GREEN, R. Communication security in internet of thing: Preventive measure and avoid ddos attack over iot network. *Proceedings of the 18th Symposium on Communications & Networking*, p. 8–15, 2015. Citado 7 vezes nas páginas 4, 9, 26, 27, 29, 39 e 40.

Apêndices

APÊNDICE A – Código do ataque

Os códigos utilizados para a criação do ataque estão apresentados a seguir:

Listagem A.1 – RPL-UDP *sink* modificado

```

1  /* -----> Destination <-----
   *
3  *
   *
5  *
   * Redistribution and use in source and binary forms , with or without
7  * modification , are permitted provided that the following conditions
   * are met:
9  * 1. Redistributions of source code must retain the above copyright
   * notice , this list of conditions and the following disclaimer .
11 * 2. Redistributions in binary form must reproduce the above copyright
   * notice , this list of conditions and the following disclaimer in the
13 * documentation and/or other materials provided with the distribution .
   * 3. Neither the name of the Institute nor the names of its contributors
15 * may be used to endorse or promote products derived from this software
   * without specific prior written permission .
17 *
   * THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ‘‘AS IS ’’
   AND
19 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
   PURPOSE
21 * ARE DISCLAIMED. IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE
   LIABLE
   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL
23 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
   * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
25 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
   STRICT
   * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
   WAY
27 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
   * SUCH DAMAGE.
29 *
   * This file is part of the Contiki operating system .
31 *
   */
33 #include "contiki.h"

```

```

35 #include "contiki-lib.h"
#include "contiki-net.h"
37 #include "net/uip.h"
#include "net/rpl/rpl.h"
39 #include "net/rime/rimeaddr.h"
#include "sys/compower.h"
41 #include "powertrace.h"
#include "sys/clock.h"
43
#include "net/netstack.h"
45 #include "dev/button-sensor.h"
#include "dev/light-sensor.h"
47 #include "dev/sht11-sensor.h"
#include "dev/radio-sensor.h"
49 #include "dev/serial-line.h"
#include "if CONTIKI_TARGET_Z1
51 #include "dev/uart0.h"
#include "else
53 #include "dev/uart1.h"
#include "endif
55 #include <stdio.h>
#include <stdlib.h>
57 #include <string.h>
#include <ctype.h>
59 #include <math.h>

61 #include "collect-common.h"
#include "collect-view.h"
63
#define DEBUG DEBUG_PRINT
65 #include "net/uip-debug.h"

67 #define UIP_IP_BUF ((struct uip_ip_hdr *)&uip_buf[UIP_LLH_LEN])

69 #define UDP_CLIENT_PORT 8775
#define UDP_SERVER_PORT 5688
71
static struct uip_udp_conn *server_conn;
73
PROCESS(udp_server_process, "UDP server process");
75 AUTOSTART_PROCESSES(&udp_server_process, &collect_common_process);
/*-----*/
77 void
collect_common_set_sink(void)
79 {
printf("I am sink!\n");
81 }

```

```
/*-----*/
83 void
collect_common_net_print(void)
85 {
    printf("I am sink!\n");
87 }
/*-----*/
89 void
collect_common_send(void)
91 {
    struct {
93         uint8_t seqno;
          uint8_t for_alignment;
95         struct collect_view_data_msg msg;
    } msg;
97
    uint16_t parent_etx;
99     uint16_t rtmetric;
    uint16_t num_neighbors;
101    uint16_t beacon_interval;
    rimeaddr_t parent;
103
    memset(&msg, 0, sizeof(msg));
105    msg.seqno = 128;

107    rimeaddr_copy(&parent, &rimeaddr_null);
    parent_etx = 0;
109    rtmetric = 0;
    beacon_interval = 0;
111    num_neighbors = 0;

113    collect_view_construct_message(&msg.msg, &parent,
                                   parent_etx, rtmetric,
115                                   num_neighbors, beacon_interval);

117 }
/*-----*/
119 void
collect_common_net_init(void)
121 {
    #if CONTIKI_TARGET_Z1
123     uart0_set_input(serial_line_input_byte);
    #else
125     uart1_set_input(serial_line_input_byte);
    #endif
127     serial_line_init();
}
```

```

129     PRINTF("I am sink!\n");
130 }
131 /*-----*/
132 static void
133 tcpip_handler(void)
134 {
135     uint8_t *appdata;
136     rimeaddr_t sender;
137     uint8_t seqno;
138     uint8_t hops;
139
140     if(uiplib_newdata()) {
141         appdata = (uint8_t *)uiplib_appdata;
142         sender.u8[0] = UIP_IP_BUF->srcipaddr.u8[15];
143         sender.u8[1] = UIP_IP_BUF->srcipaddr.u8[14];
144         seqno = *appdata;
145         hops = uip_ds6_if.cur_hop_limit - UIP_IP_BUF->tll + 1;
146         collect_common_recv(&sender, seqno, hops,
147                             appdata + 2, uip_datalen() - 2);
148     }
149 }
150 /*-----*/
151 static void
152 print_local_addresses(void)
153 {
154     int i;
155     uint8_t state;
156
157     PRINTF("Server IPv6 addresses: ");
158     for(i = 0; i < UIP_DS6_ADDR_NB; i++) {
159         state = uip_ds6_if.addr_list[i].state;
160         if(state == ADDR_TENTATIVE || state == ADDR_PREFERRED) {
161             PRINT6ADDR(&uip_ds6_if.addr_list[i].ipaddr);
162             PRINTF("\n");
163             /* hack to make address "final" */
164             if (state == ADDR_TENTATIVE) {
165                 uip_ds6_if.addr_list[i].state = ADDR_PREFERRED;
166             }
167         }
168     }
169 }
170 /*-----*/
171 PROCESS_THREAD(udp_server_process, ev, data)
172 {
173     uip_ipaddr_t ipaddr;
174     struct uip_ds6_addr *root_if;
175

```

```

177  PROCESS_BEGIN();
178  PROCESS_PAUSE();

179  SENSORS_ACTIVATE(button_sensor);

181  PRINTF("UDP server started\n");

183  #if UIP_CONF_ROUTER
    uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 1);
185  /* uip_ds6_set_addr_iid(&ipaddr, &uip_lladdr); */
    uip_ds6_addr_add(&ipaddr, 0, ADDR_MANUAL);
187    root_if = uip_ds6_addr_lookup(&ipaddr);
    if (root_if != NULL) {
189        rpl_dag_t *dag;
        dag = rpl_set_root(RPL_DEFAULT_INSTANCE, (uip_ip6addr_t *)&ipaddr);
191        uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0);
        rpl_set_prefix(dag, &ipaddr, 64);
193        PRINTF("created a new RPL dag\n");
    } else {
195        PRINTF("failed to create a new RPL DAG\n");
    }
197  #endif /* UIP_CONF_ROUTER */

199  print_local_addresses();

201  /* The data sink runs with a 100% duty cycle in order to ensure high
    packet reception rates. */
203  NETSTACK_RDC.off(1);

205  server_conn = udp_new(NULL, UIP_HTONS(UDP_CLIENT_PORT), NULL);
    udp_bind(server_conn, UIP_HTONS(UDP_SERVER_PORT));

207

209  PRINTF("Created a server connection with remote address ");
    PRINT6ADDR(&server_conn->ripaddr);
    PRINTF(" local/remote port %u/%u\n", UIP_HTONS(server_conn->lport),
211          UIP_HTONS(server_conn->rport));

213  while(1) {
    PROCESS_YIELD();
215    if (ev == tcpip_event) {
        tcpip_handler();
217    } else if (ev == sensors_event && data == &button_sensor) {
        PRINTF("Initiaing global repair\n");
219        rpl_repair_root(RPL_DEFAULT_INSTANCE);
    }
221  }

```

```

223 PROCESS_END();
    }

```

Listagem A.2 – RPL-UDP *sender* modificado

```

/* -----> Source <-----
2  *
3  *
4  *
5  *
6  * Redistribution and use in source and binary forms, with or without
7  * modification, are permitted provided that the following conditions
8  * are met:
9  * 1. Redistributions of source code must retain the above copyright
10 * notice, this list of conditions and the following disclaimer.
11 * 2. Redistributions in binary form must reproduce the above copyright
12 * notice, this list of conditions and the following disclaimer in the
13 * documentation and/or other materials provided with the distribution.
14 * 3. Neither the name of the Institute nor the names of its contributors
15 * may be used to endorse or promote products derived from this software
16 * without specific prior written permission.
17 *
18 * THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ‘‘AS IS’’
19 * AND
20 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
22 * PURPOSE
23 * ARE DISCLAIMED. IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE
24 * LIABLE
25 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
26 * CONSEQUENTIAL
27 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
28 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
29 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
30 * STRICT
31 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
32 * WAY
33 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
34 * SUCH DAMAGE.
35 *
36 * This file is part of the Contiki operating system.
37 *
38 */

#include "contiki.h"
#include "net/uip.h"
#include "net/uip-ds6.h"
#include "net/uip-udp-packet.h"

```

```

38 #include "net/neighbor-info.h"
   #include "net/rpl/rpl.h"
40 #include "dev/serial-line.h"
   #if CONTIKI_TARGET_Z1
42 #include "dev/uart0.h"
   #else
44 #include "dev/uart1.h"
   #endif
46 #include "collect-common.h"
   #include "collect-view.h"
48
50 #include <stdio.h>
   #include <string.h>

52 #define UDP_CLIENT_PORT 8775
   #define UDP_SERVER_PORT 5688
54
56 #define DEBUG DEBUG_PRINT
   #include "net/ui-debug.h"

58
60 #include "lib/random.h"
   #include "sys/ctimer.h"
   #include "dev/button-sensor.h"
62
64 #include "powertrace.h"

66
68 #ifndef POWERTRACE_H
   #define POWERTRACE_H

70 void powertrace_start(CLOCK_SECOND * 5);
   void powertrace_stop(void);
72 void powertrace_print(char *str);

74 #endif /* POWERTRACE_H */

76 static struct uip_udp_conn *client_conn;
   static uip_ipaddr_t server_ipaddr;
78 int attackInterval = 1; //packet transmission interval when attacking
   int normalInterval = 60; //packet transmission interval when not attacking
80
82 void minha_funcao(void);

   /*-----*/
84 PROCESS(udp_client_process, "UDP client process");

```

```

AUTOSTART_PROCESSES(&udp_client_process , &collect_common_process);
86 /*-----*/
void
88 collect_common_set_sink(void)
{
90     /* A udp client can never become sink */
}
92 /*-----*/
extern uip_ds6_route_t uip_ds6_routing_table[UIP_DS6_ROUTE_NB];
94
void
96 collect_common_net_print(void)
{
98     rpl_dag_t *dag;
    int i;
100     /* Let's suppose we have only one instance */
    dag = rpl_get_any_dag();
102     if(dag->preferred_parent != NULL) {
        PRINTF("Preferred parent: ");
104         PRINT6ADDR(&dag->preferred_parent->addr);
        PRINTF("\n");
106     }
    PRINTF("Route entries:\n");
108     for(i = 0; i < UIP_DS6_ROUTE_NB; i++) {
        if(uip_ds6_routing_table[i].isused) {
110             PRINT6ADDR(&uip_ds6_routing_table[i].ipaddr);
            PRINTF("\n");
112         }
    }
114     PRINTF("----\n");
}
116 /*-----*/
static void
118 tcpip_handler(void)
{
120     if(uip_newdata()) {
        /* Ignore incoming data */
122     }
}
124 /*-----*/
static void
126 send_packet(void *ptr)
{
128     minha_funcao();
}
130 /*-----*/
void

```

```

132 collect_common_send(void)
133 {
134 }
135 void
136 minha_funcao(void)
137 {
138     static uint8_t seqno;
139     struct {
140         uint8_t seqno;
141         uint8_t for_alignment;
142         struct collect_view_data_msg msg;
143     } msg;
144     /* struct collect_neighbor *n; */
145     uint16_t parent_etx;
146     uint16_t rtmetric;
147     uint16_t num_neighbors;
148     uint16_t beacon_interval;
149     rpl_parent_t *preferred_parent;
150     rimeaddr_t parent;
151     rpl_dag_t *dag;
152
153     if(client_conn == NULL) {
154         /* Not setup yet */
155         return;
156     }
157     memset(&msg, 0, sizeof(msg));
158     seqno++;
159     if(seqno == 0) {
160         /* Wrap to 128 to identify restarts */
161         seqno = 128;
162     }
163     msg.seqno = seqno;
164
165     rimeaddr_copy(&parent, &rimeaddr_null);
166     parent_etx = 0;
167
168     /* Let's suppose we have only one instance */
169     dag = rpl_get_any_dag();
170     if(dag != NULL) {
171         preferred_parent = dag->preferred_parent;
172         if(preferred_parent != NULL) {
173             uip_ds6_nbr_t *nbr;
174             nbr = uip_ds6_nbr_lookup(&preferred_parent->addr);
175             if(nbr != NULL) {
176                 /* Use parts of the IPv6 address as the parent address, in reversed
177                  byte order. */
178                 parent.u8[RIMEADDR_SIZE - 1] = nbr->ipaddr.u8[sizeof(uip_ipaddr_t)

```

```

- 2];
178     parent.u8[RIMEADDR_SIZE - 2] = nbr->ipaddr.u8[sizeof(uip_ipaddr_t)
- 1];
    parent_etx = neighbor_info_get_metric((rimeaddr_t *) &nbr->lladdr)
/ 2;
180     }
    }
182     rtmetric = dag->rank;
    beacon_interval = (uint16_t) ((2L << dag->instance->dio_intcurrent) /
1000);
184     num_neighbors = RPL_PARENT_COUNT(dag);
} else {
186     rtmetric = 0;
    beacon_interval = 0;
188     num_neighbors = 0;
}

190
/* num_neighbors = collect_neighbor_list_num(&tc.neighbor_list); */
192 collect_view_construct_message(&msg.msg, &parent,
                                parent_etx, rtmetric,
194                                num_neighbors, beacon_interval);

196 uip_udp_packet_sendto(client_conn, &msg, sizeof(msg),
                        &server_ipaddr, UIP_HTONS(UDP_SERVER_PORT));
198 }
/*-----*/
200 void
collect_common_net_init(void)
202 {
    #if CONTIKI_TARGET_Z1
204     uart0_set_input(serial_line_input_byte);
    #else
206     uart1_set_input(serial_line_input_byte);
    #endif
208     serial_line_init();
}
/*-----*/
210
static void
212 print_local_addresses(void)
{
214     int i;
    uint8_t state;

216
    for(i = 0; i < UIP_DS6_ADDR_NB; i++) {
218         state = uip_ds6_if.addr_list[i].state;
        if(uip_ds6_if.addr_list[i].isused &&
220            (state == ADDR_TENTATIVE || state == ADDR_PREFERRED)) {

```

```

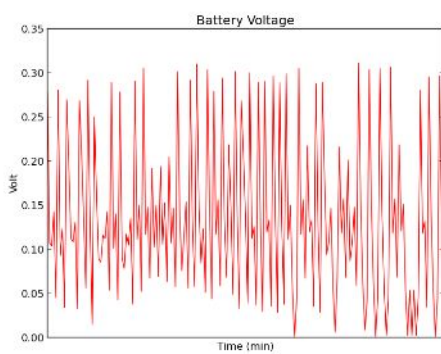
222     PRINT6ADDR(&uip_ds6_if.addr_list[i].ipaddr);
223     PRINTF("\n");
224     /* hack to make address "final" */
225     if (state == ADDR_TENTATIVE) {
226         uip_ds6_if.addr_list[i].state = ADDR_PREFERRED;
227     }
228 }
229 }
230 /*-----*/
231 static void
232 set_global_address(void)
233 {
234     uip_ipaddr_t ipaddr;
235
236     uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0);
237     uip_ds6_set_addr_iid(&ipaddr, &uip_lladdr);
238     uip_ds6_addr_add(&ipaddr, 0, ADDR_AUTOCONF);
239
240     /* set server address */
241     uip_ip6addr(&server_ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 1);
242 }
243 /*-----*/
244 PROCESS_THREAD(udp_client_process, ev, data)
245 {
246     static struct etimer periodic;
247     static struct ctimer backoff_timer;
248
249     PROCESS_BEGIN();
250     SENSORS_ACTIVATE(button_sensor); //activate right button sensor. Used to
251                                     start an attack
252     PROCESS_PAUSE();
253
254     set_global_address();
255
256     PRINTF("UDP client process started\n");
257
258     print_local_addresses();
259
260     /* new connection with remote host */
261     client_conn = udp_new(NULL, UIP_HTONS(UDP_SERVER_PORT), NULL);
262     udp_bind(client_conn, UIP_HTONS(UDP_CLIENT_PORT));
263
264     PRINTF("Created a connection with the server ");
265     PRINT6ADDR(&client_conn->ripaddr);
266     PRINTF(" local/remote port %u/%u\n",

```

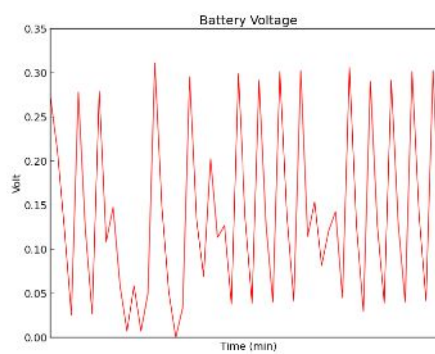
```
268     UIP_HTONS(client_conn->lport), UIP_HTONS(client_conn->rport));
269
270     etimer_set(&periodic, normalInterval * CLOCK_SECOND); //set an event
    timer
271 while(1) {
272     PROCESS_YIELD();
273
274     if(etimer_expired(&periodic)) { //Check if an event timer has expired.
        etimer_reset(&periodic); //Reset an event timer with the same
        interval as was previously set
        etimer_set(&backoff_timer, (random_rand() % (normalInterval *
        CLOCK_SECOND)), send_packet, NULL); //This function is used to set a
        callback timer for a time sometime in the future. When the callback
        timer (backoff_timer) expires ((random_rand() % (normalInterval *
        CLOCK_SECOND)) is the interval before de timer expires), the callback
        function (send_packet) will be called with ptr (NULL) as argument.
275     }
276
277     if(ev == tcpip_event) {
278         tcpip_handler();
279     }
280     else if (ev == sensors_event && data == &button_sensor) {
281         PRINTF("SISTEMA: Iniciando Ataque\n");
282         normalInterval=attackInterval;
283         etimer_set(&periodic, normalInterval * CLOCK_SECOND);
284     }
285 }
286
287 PROCESS_END();
288 }
289 /*-----*/
```

APÊNDICE B – Simulações extras dos cenários utilizados

Além da simulação apresentada no capítulo 4, duas outras foram desenvolvidas. Seus resultados estão a seguir:

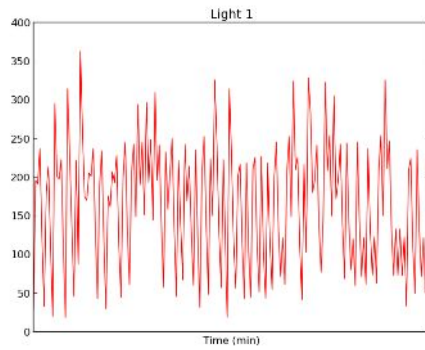


(a) Battery Voltage - antes do ataque

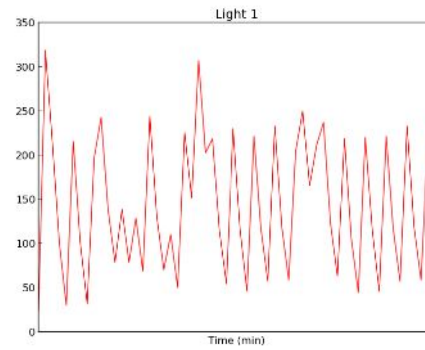


(b) Battery Voltage - durante o ataque

Figura 35 – Execução 2 - Cenário 1: Battery Voltage antes e durante o ataque

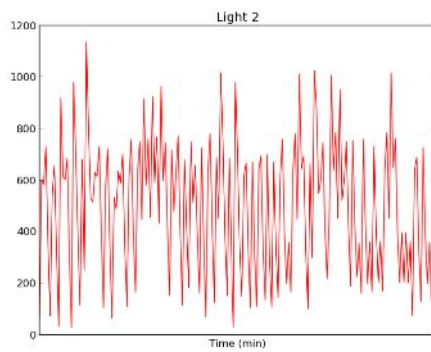


(a) Light 1 - antes do ataque

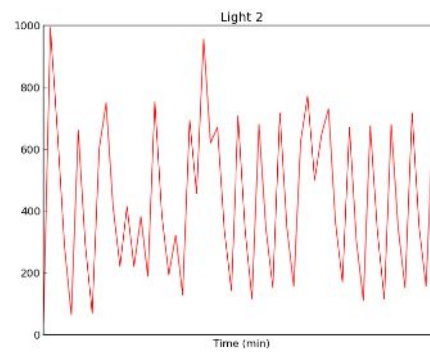


(b) Light 1 - durante o ataque

Figura 36 – Execução 2 - Cenário 1: Light 1 antes e durante o ataque

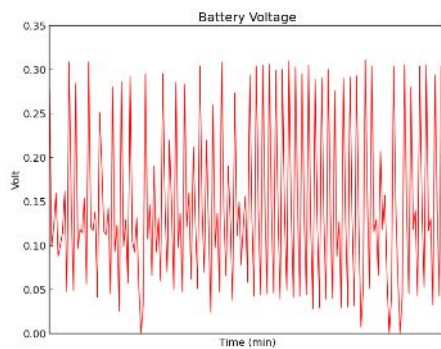


(a) Light 2 - antes do ataque

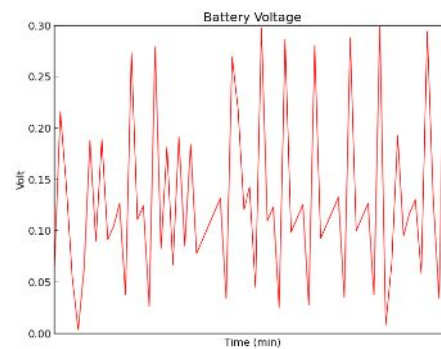


(b) Light 2 - durante o ataque

Figura 37 – Execução 2 - Cenário 1: Light 2 antes e durante o ataque

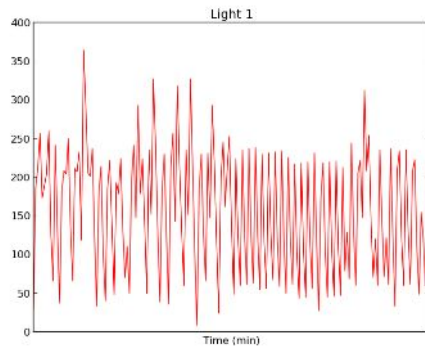


(a) Battery Voltage - antes do ataque

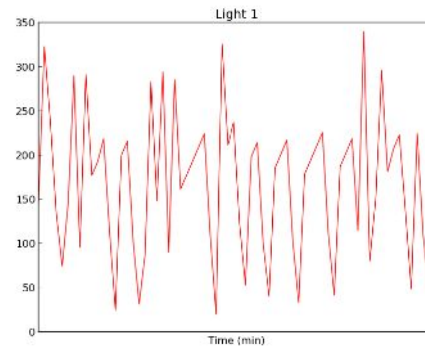


(b) Battery Voltage - durante o ataque

Figura 38 – Execução 2 - Cenário 2: Battery Voltage antes e durante o ataque

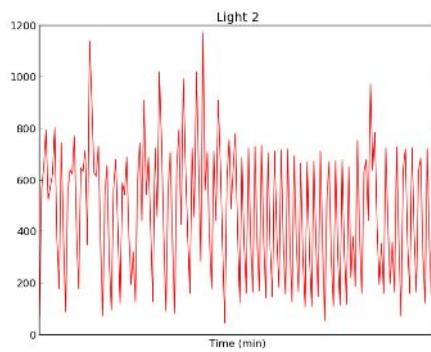


(a) Light 1 - antes do ataque

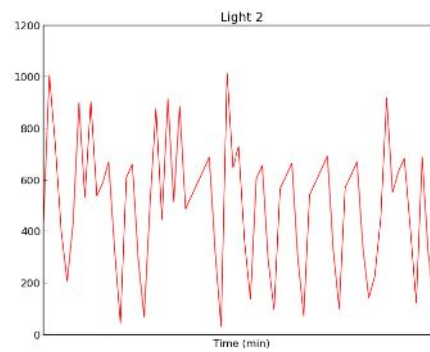


(b) Light 1 - durante o ataque

Figura 39 – Execução 2 - Cenário 2: Light 1 antes e durante o ataque

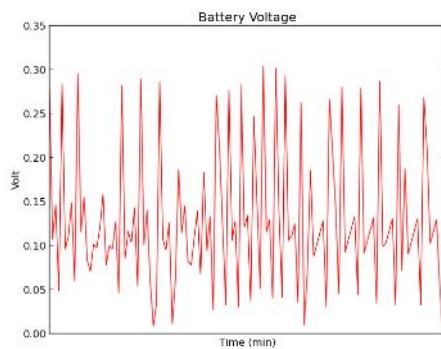


(a) Light 2 - antes do ataque

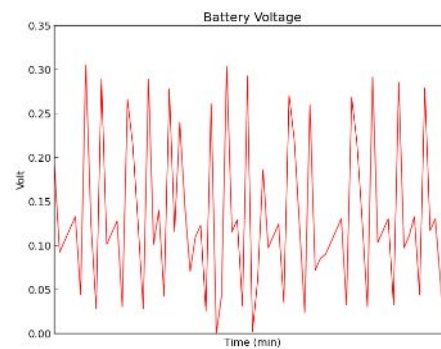


(b) Light 2 - durante o ataque

Figura 40 – Execução 2 - Cenário 2: Light 2 antes e durante o ataque

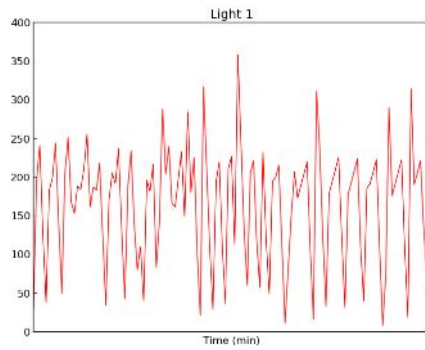


(a) Battery Voltage - antes do ataque

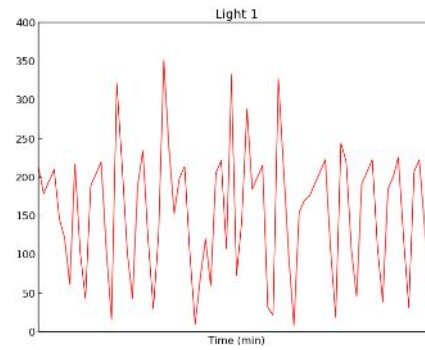


(b) Battery Voltage - durante o ataque

Figura 41 – Execução 2 - Cenário 3: Battery Voltage antes e durante o ataque

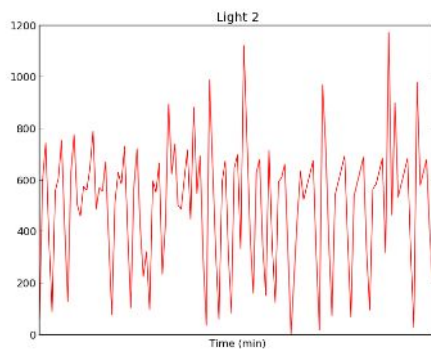


(a) Light 1 - antes do ataque

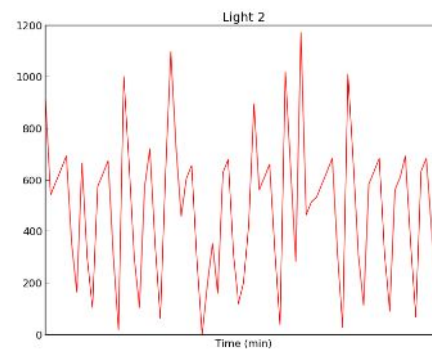


(b) Light 1 - durante o ataque

Figura 42 – Execução 2 - Cenário 3: Light 1 antes e durante o ataque

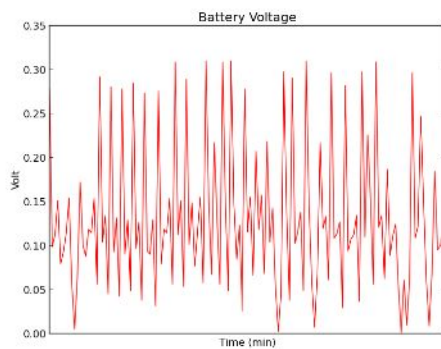


(a) Light 2 - antes do ataque

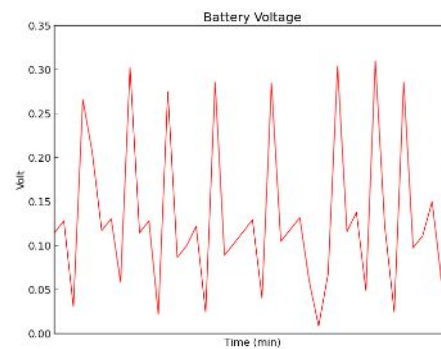


(b) Light 2 - durante o ataque

Figura 43 – Execução 2 - Cenário 3: Light 2 antes e durante o ataque

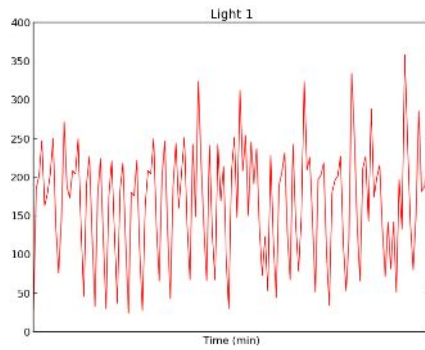


(a) Battery Voltage - antes do ataque

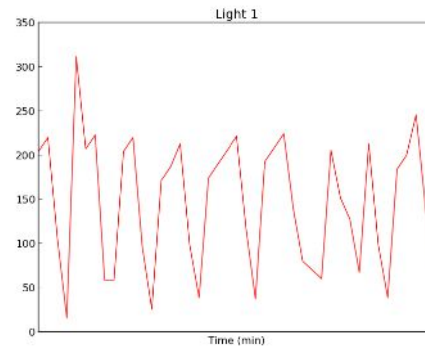


(b) Battery Voltage - durante o ataque

Figura 44 – Execução 2 - Cenário 4: Battery Voltage antes e durante o ataque

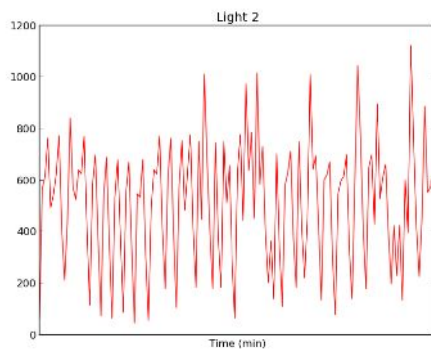


(a) Light 1 - antes do ataque

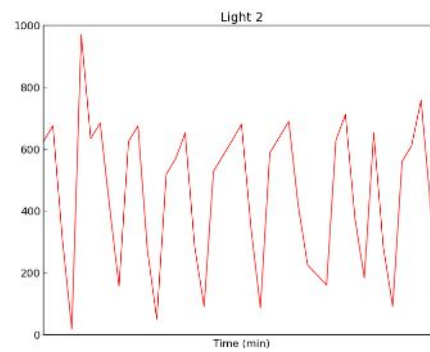


(b) Light 1 - durante o ataque

Figura 45 – Execução 2 - Cenário 4: Light 1 antes e durante o ataque

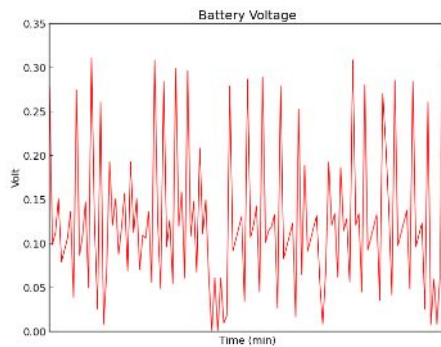


(a) Light 2 - antes do ataque

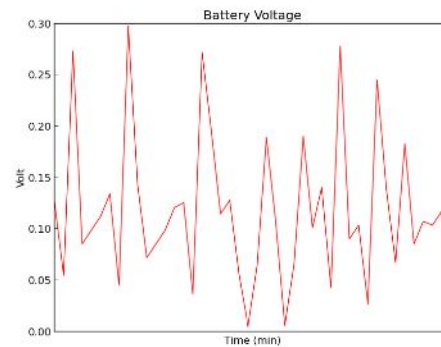


(b) Light 2 - durante o ataque

Figura 46 – Execução 2 - Cenário 4: Light 2 antes e durante o ataque

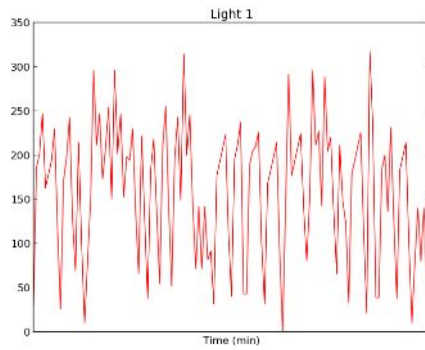


(a) Battery Voltage - antes do ataque

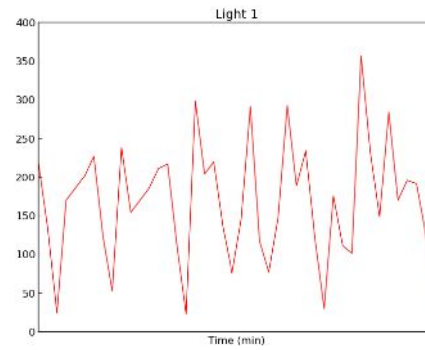


(b) Battery Voltage - durante o ataque

Figura 47 – Execução 2 - Cenário 5: Battery Voltage antes e durante o ataque

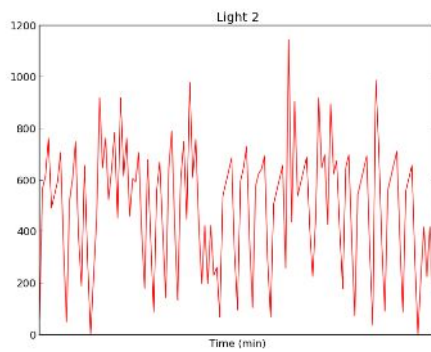


(a) Light 1 - antes do ataque

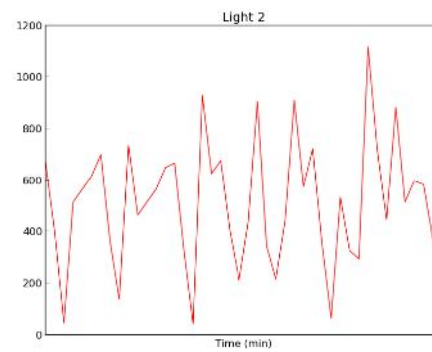


(b) Light 1 - durante o ataque

Figura 48 – Execução 2 - Cenário 5: Light 1 antes e durante o ataque

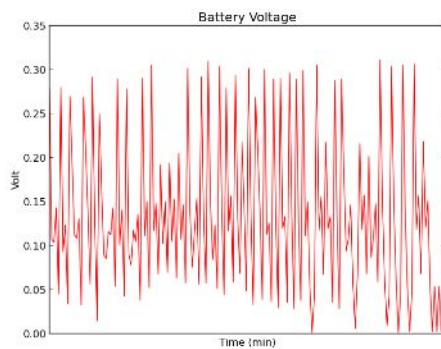


(a) Light 2 - antes do ataque

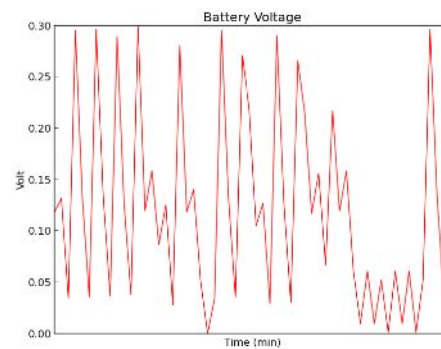


(b) Light 2 - durante o ataque

Figura 49 – Execução 2 - Cenário 5: Light 2 antes e durante o ataque

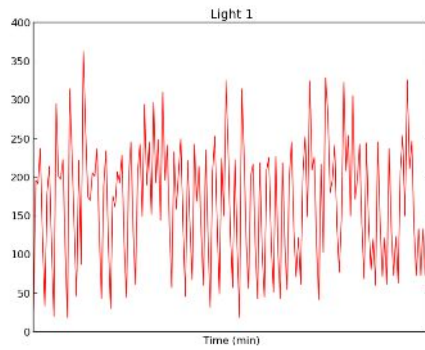


(a) Battery Voltage - antes do ataque

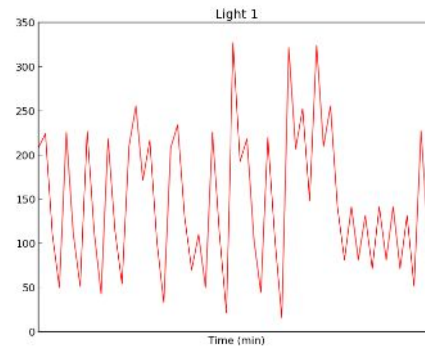


(b) Battery Voltage - durante o ataque

Figura 50 – Execução 3 - Cenário 1: Battery Voltage antes e durante o ataque

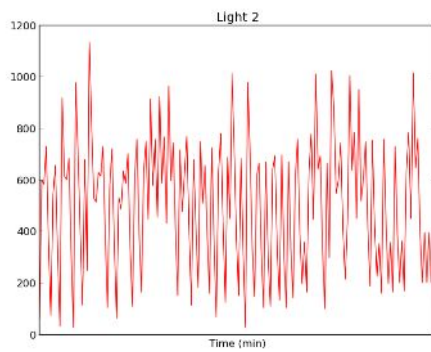


(a) Light 1 - antes do ataque

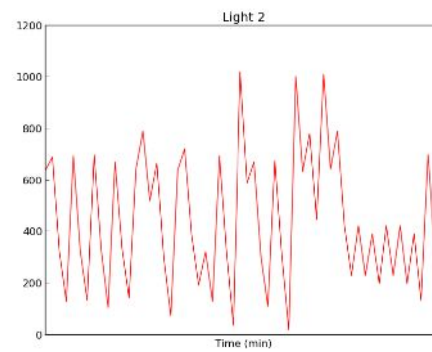


(b) Light 1 - durante o ataque

Figura 51 – Execução 3 - Cenário 1: Light 1 antes e durante o ataque

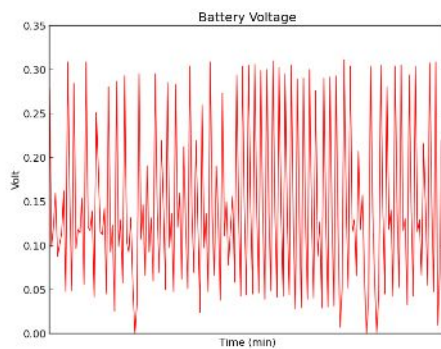


(a) Light 2 - antes do ataque

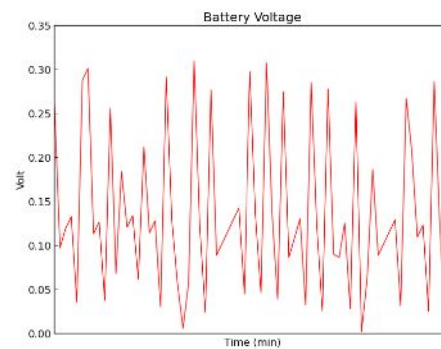


(b) Light 2 - durante o ataque

Figura 52 – Execução 3 - Cenário 1: Light 2 antes e durante o ataque

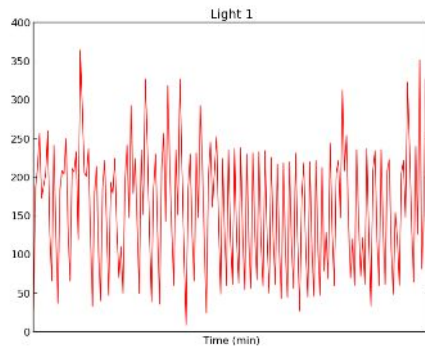


(a) Battery Voltage - antes do ataque

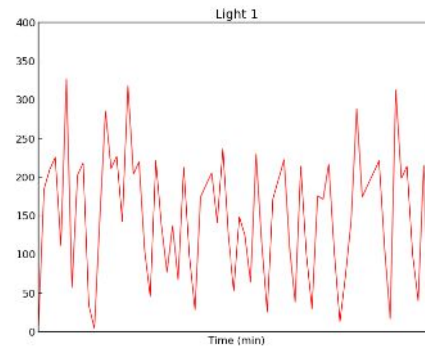


(b) Battery Voltage - durante o ataque

Figura 53 – Execução 3 - Cenário 2: Battery Voltage antes e durante o ataque

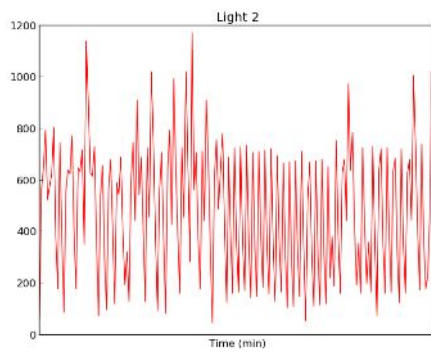


(a) Light 1 - antes do ataque

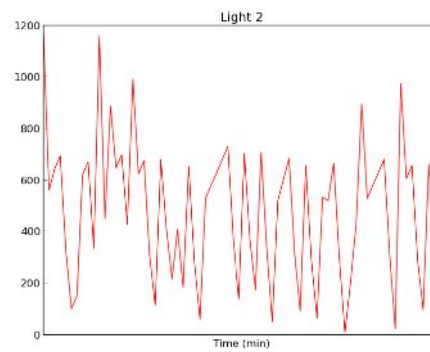


(b) Light 1 - durante o ataque

Figura 54 – Execução 3 - Cenário 2: Light 1 antes e durante o ataque

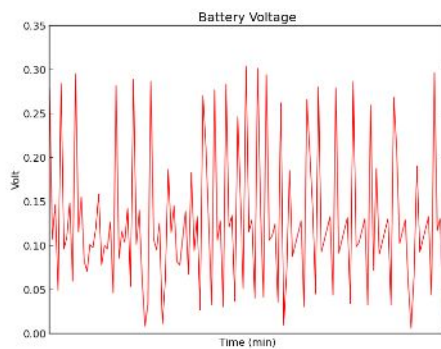


(a) Light 2 - antes do ataque

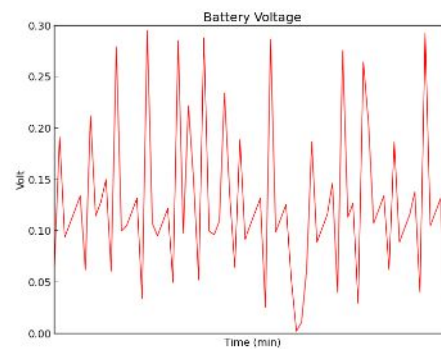


(b) Light 2 - durante o ataque

Figura 55 – Execução 3 - Cenário 2: Light 2 antes e durante o ataque

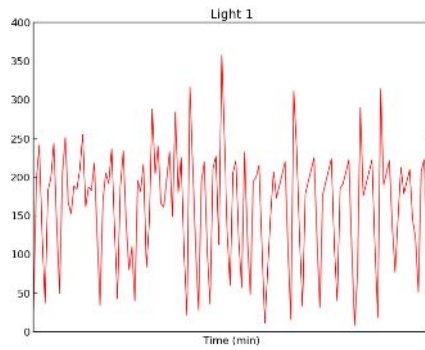


(a) Battery Voltage - antes do ataque

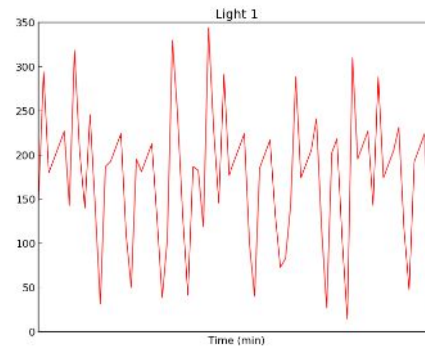


(b) Battery Voltage - durante o ataque

Figura 56 – Execução 3 - Cenário 3: Battery Voltage antes e durante o ataque

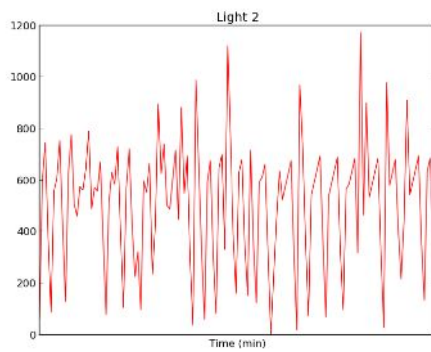


(a) Light 1 - antes do ataque

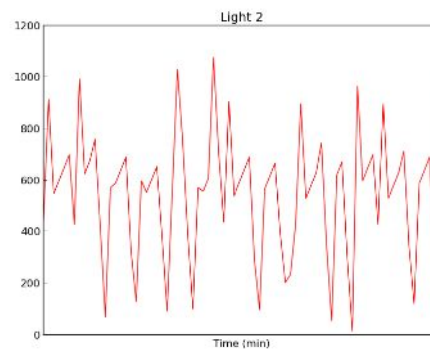


(b) Light 1 - durante o ataque

Figura 57 – Execução 3 - Cenário 3: Light 1 antes e durante o ataque

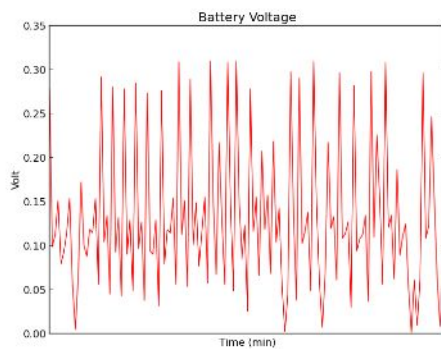


(a) Light 2 - antes do ataque

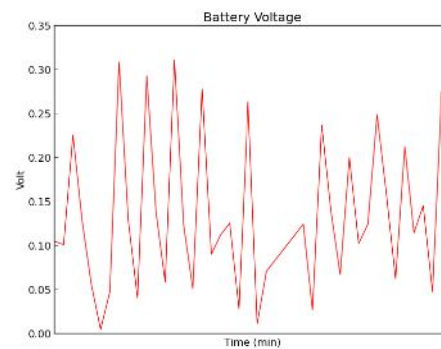


(b) Light 2 - durante o ataque

Figura 58 – Execução 3 - Cenário 3: Light 2 antes e durante o ataque

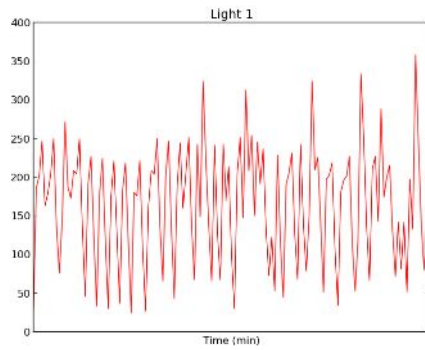


(a) Battery Voltage - antes do ataque

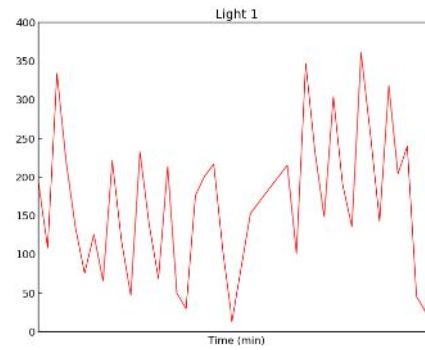


(b) Battery Voltage - durante o ataque

Figura 59 – Execução 3 - Cenário 4: Battery Voltage antes e durante o ataque

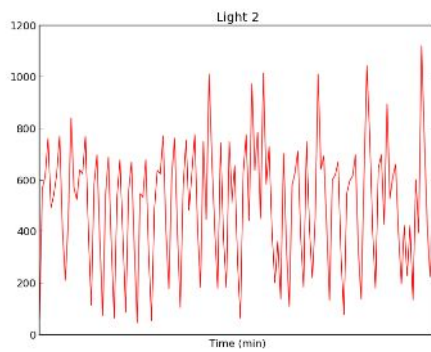


(a) Light 1 - antes do ataque

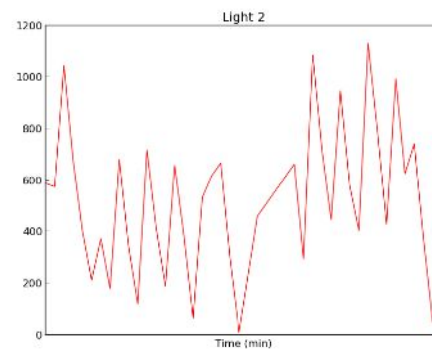


(b) Light 1 - durante o ataque

Figura 60 – Execução 3 - Cenário 4: Light 1 antes e durante o ataque

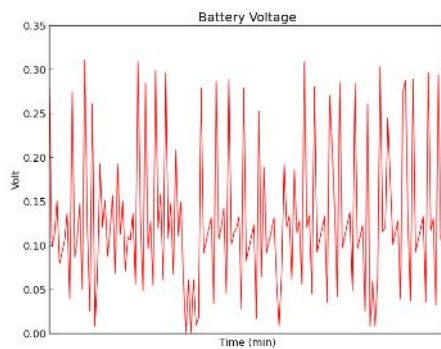


(a) Light 2 - antes do ataque

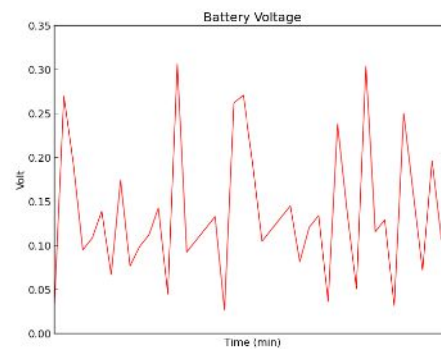


(b) Light 2 - durante o ataque

Figura 61 – Execução 3 - Cenário 4: Light 2 antes e durante o ataque

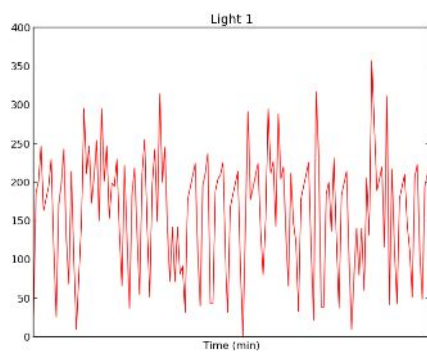


(a) Battery Voltage - antes do ataque

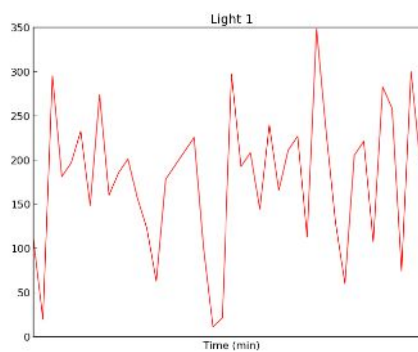


(b) Battery Voltage - durante o ataque

Figura 62 – Execução 3 - Cenário 5: Battery Voltage antes e durante o ataque

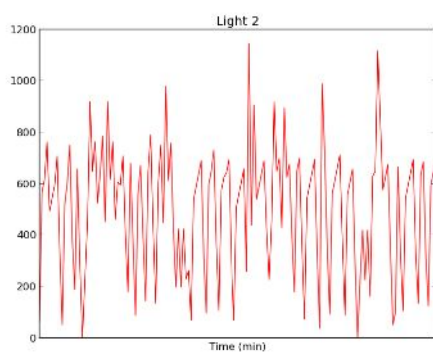


(a) Light 1 - antes do ataque

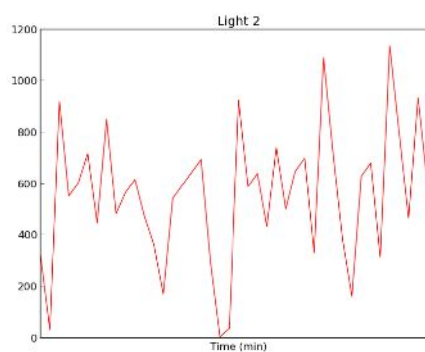


(b) Light 1 - durante o ataque

Figura 63 – Execução 3 - Cenário 5: Light 1 antes e durante o ataque



(a) Light 2 - antes do ataque



(b) Light 2 - durante o ataque

Figura 64 – Execução 3 - Cenário 5: Light 2 antes e durante o ataque