SBRC 2019

Gramado | RS

XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

6 a 10 de maio

Gramado | RS

ANAIS DO WSCDC 2019

sbrc2019.sbc.org.br

XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

6 a 10 de maio

Gramado | RS

ANAIS DO WSCDC 2019

sbrc2019.sbc.org.br

EDITORA
Sociedade Brasileira de Computação (SBC)

ORGANIZAÇÃO
Alberto Egon Schaeffer Filho (UFRGS)
Weverton Luis da Costa Cordeiro (UFRGS)
Stênio Fernandes (UFPE)
Cesar Marcondes (ITA)
Lourenço Pereira (GoDigital)
Paulo Sergio Melo de Carvalho (ANSP)
Pedro Rosa (UFU)
Rodrigo Miani (UNICAMP)
Paulo Matias (UFSCar)

REALIZAÇÃO
Sociedade Brasileira de Computação (SBC)
Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
Laboratório Nacional de Redes de Computadores (LARC)

Copyright ©2019 da Sociedade Brasileira de Computação
Todos os direitos reservados

Capa: Dórian Fogliatto
Produção Editorial: Carlos Raniery P. dos Santos (UFSM)

Cópias Adicionais:
Sociedade Brasileira de Computação (SBC)
Av. Bento Gonçalves, 9500- Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia - CEP 91.509-900 - Porto Alegre - RS
Fone: (51) 3308-6835
E-mail: sbc@sbc.org.br

II Workshop de Segurança Cibernética em Dispositivos Conectados (2: 2019: Gramado, RS).

Anais / II Workshop de Segurança Cibernética em Dispositivos Conectados - WSCDC; organizado por Alberto Egon Schaeffer Filho, Weverton Luis da Costa Cordeiro, Stênio Fernandes, Cesar Marcondes, Lourenço Pereira, Paulo Sergio Melo de Carvalho, Pedro Rosa, Rodrigo Miani, Paulo Matias - Porto Alegre: SBC, 2019

104 p. il. 21 cm.
Vários autores
Inclui bibliografias

1. Redes de Computadores. 2. Sistemas Distribuídos. I. Schaeffer-Filho, Alberto Egon II. Cordeiro, Weverton Luis da Costa III. Fernandes, Stênio IV. Marcondes, Cesar V. Pereira, Lourenço VI. Carvalho, Paulo Sergio Melo VII. Rosa, Pedro VIII Miani, Rodrigo IV. Matias, Paulo VI Título.

Sociedade Brasileira de Computação

Presidência
Lisandro Zambenedetti Granville (UFRGS), Presidente
Thais Vasconcelos Batista (UFRN),Vice-Presidente

Diretorias
Renata de Matos Galante (UFGRS), Diretora Administrativa
Carlos André Guimarães Ferraz (UFPE), Diretor de Finanças
Antônio Jorge Gomes Abelém (UFPA), Diretor de Eventos e Comissões Especiais
Renata Mendes de Araujo (UPM), Diretora de Educação
José Viterbo Filho (UFF), Diretor de Publicações
Claudia Lage Rebello da Motta (UFRJ), Diretora de Planejamento e Programas Especiais
Marcelo Duduchi Feitosa (CEETEPS), Diretor de Secretarias Regionais
Eliana Almeida (UFAL), Diretora de Divulgação e Marketing
Ricardo de Oliveira Anido (UNICAMP), Diretor de Relações Profissionais
Esther Colombini (UNICAMP), Diretora de Competições Científicas
Raimundo José de Araújo Macêdo (UFBA), Diretor de Cooperação com Sociedades Científicas
Cláudia Cappeli (UNIRIO), Diretora de Articulação com Empresas

Contato
Av. Bento Gonçalves, 9500
Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia
91.509-900 – Porto Alegre RS
CNPJ: 29.532.264/0001-78
http://www.sbc.org.br

Laboratório Nacional de Redes de Computadores (LARC)

Diretor do Conselho Técnico-Científico
Paulo André da Silva Gonçalves (UFPE)

Vice-Diretora do Conselho Técnico-Científico
Rossana Maria de Castro Andrade (UFC)

Diretor Executivo
Ronaldo Alves Ferreira (UFMS)

Vice-Diretor Executivo
Danielo Gonçalves Gomes (UFC)

Membros Institucionais
SESU/MEC, INPE/MCT, UFRGS, UFMG, UFPE, UFCG (ex-UFPB Campus Campina Grande), UFRJ, USP, PUC-Rio, UNICAMP, LNCC, IME, UFSC, UTFPR, UFC, UFF, UFSCar, IFCE (CEFET-CE), UFRN, UFES, UFBA, UNIFACS, UECE, UFPR, UFPA, UFAM, UFABC, PUCPR, UFMS, UnB, PUC-RS, PUCMG, UNIRIO, UFS e UFU.

Contato
Universidade Federal de Mato Grosso do Sul (UFMS)
Faculdade de Computação
Caixa Postal 549
CEP 79.070-900 Campo Grande - MS - Brasil
http://www.larc.org.br

Organização do SBRC 2019

Coordenadores Gerais
Alberto Egon Schaeffer Filho (UFRGS)
Weverton Luis da Costa Cordeiro (UFRGS)

Coordenadores do Comitê de Programa
Antônio Jorge Gomes Abelém (UFPA)
Fabíola Gonçalves Pereira Greve (UFBA)

Coordenador de Palestras e Tutoriais
Ítalo Cunha (UFMG)

Coordenador de Painéis e Debates
Artur Ziviani (LNCC)

Coordenador de Minicursos
Miguel Elias Mitre Campista (UFRJ)

Coordenador de Workshops
Stênio Fernandes (UFPE)

Coordenador do Salão de Ferramentas
Leandro Villas (UNICAMP)

Coordenador do Concurso de Teses e Dissertações
Daniel Fernandes Macedo (UFMG)

Coordenadores do Hackaton
Raquel Lopes (UFMG)
Luis Carlos de Bona (UFPR)

Comitê de Organização Local
Avelino Zorzo (PUCRS)
Carlos Raniery Paula dos Santos (UFSM)
Cristiano Bonato Both (Unisinos)
Guilherme Rodrigues (IFSUL Charqueadas)
Jéferson Campos Nobre (UFRGS)
Juliano Wickboldt (UFRGS)
Marcelo Caggiani Luizelli (Unipampa)
Marcelo da Silva Conterato (PUCRS)
Rafael Pereira Esteves (IFRS)
Rodrigo Mansilha (Unipampa)
Tiago Ferreto (PUCRS)
Vinícius Guimarães (IFSUL Charquadas)

Comite Consultivo
Fabíola Gonçalves Pereira Greve (UFBA)
Paulo André da Silva Gonçalves (UFPE)
Fábio Luciano Verdi (UFSCAR)
Jó Ueyama (USP)
Antônio Jorge Gomes Abelém (UFPA)
Eduardo Cerqueira (UFPA)
Luiz Fernando Bittencourt (Unicamp)
Rossana Andrade (UFC)
Michele Nogueira (UFPR)
Edmundo Madeira (UNICAMP)

Mensagem dos Coordenadores Gerais

É com grande alegria e orgulho que, após 9 anos, estamos trazendo o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) novamente para a cidade de Gramado. O SBRC 2019 acontece em um ano que é especial por várias razões: primeiro, o aniversário de 50 anos da ARPANET, a primeira rede a implementar o conjunto de protocolos TCP/IP, que acabaram se estabelecendo mais tarde como os fundamentos da Internet; os 30 anos da invenção da World Wide Web pelo cientista Inglês Tim Berners-Lee; a comemoração dos 30 anos do .br (country code top-level domain), o qual tem atualmente mais de 4 milhões de domínios registrados; por fim, nesse ano comemora-se os 30 anos da Rede Nacional de Ensino e Pesquisa (RNP), que iniciou em 1989 como um projeto de pesquisa. É, portanto, um imenso privilégio termos a oportunidade de sediar o SBRC em um ano que carrega tanto significado para a área de redes de computadores e sistemas distribuídos.

A 37a edição do evento se apoia no histórico de sucesso do SBRC, que tradicionalmente inclui sessões técnicas, minicursos, painéis e debates, workshops, salão de ferramentas e palestras. Além de contar novamente com um Hackathon e com um Concurso de Teses e Dissertações, teremos muitas novidades no SBRC 2019, incluindo as reuniões de mentoria, o evento MUSAS e o workshop de estudantes latino-americanos. As reuniões de mentoria permitem oportunizar conversas 1-a-1 entre estudantes de pós-graduação e pesquisadores de excelência do Brasil e do exterior. O MUSAS (MUlheres em redeS de computadores e sistemAs diStribuídos) visa fomentar conexões entre mulheres atuando nas áreas de redes de computadores e de sistemas distribuídos, no Brasil e no exterior. Já o workshop de estudantes latino-americanos (LANCOMM) serve como um ponto de encontro para estudantes da região apresentarem e discutirem o andamento de suas pesquisas, e obterem feedback construtivo de pesquisadores estrangeiros de grande prestígio.

Neste ano, a trilha principal do SBRC recebeu 228 submissões completas e que entraram no processo de avaliação. Todos os artigos receberam pelo menos 3 revisões, e após um rigoroso processo seletivo, 80 artigos foram aceitos para publicação e organizados em 23 sessões técnicas. O Salão de Ferramentas selecionou 12 trabalhos que a serem demonstrados ao longo do SBRC 2019. Por sua vez, o Concurso de Teses e Dissertações selecionou 8 dissertações de mestrado e 8 teses de doutorado para apresentação durante o evento. Além disso, esse ano o evento conta com 4 palestrantes internacionais, 3 painéis, 1 tutorial, 5 minicursos e 11 workshops.

A organização de um evento com o porte do SBRC é um processo longo, que demanda muita energia e empenho. O evento desse ano só foi possível graças ao apoio e suporte incondicional de muitos grupos de pessoas e instituições. Agradecemos em particular o apoio da SBC, do LARC, do Comitê Consultivo do SBRC e da Comissão Especial de Redes de Computadores e Sistemas Distribuídos da SBC. O evento contou com o apoio do Comitê Gestor da Internet no Brasil (CGI.br) e do Núcleo de Informação e Coordenação do Ponto BR (NIC.br), da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), do Conselho Nacional de Desenvolvimento

Científico e Tecnológico (CNPq), da Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) e da ACM SIGCOMM. Contamos novamente também com o apoio institucional da Rede Nacional de Ensino e Pesquisa (RNP). Além disso, o SBRC 2019 recebeu o apoio de importantes patrocinadores, incluindo SAP, HUAWEI, DATACOM, Google, Bedu.tech, BRDigital e AdylNet. Também agradecemos à Universidade Federal do Rio Grande do Sul (UFRGS), e em especial ao Instituto de Informática (INF), pelo suporte incondicional.

Agradecemos a todas as pessoas que trabalharam diretamente na organização do evento. Obrigado pelo excelente trabalho de todos os membros do comitê de organização: Antônio Jorge Gomes Abelém/UFPA e Fabíola Gonçalves Pereira Greve/UFBA (Coordenadores do Comitê de Programa), Stênio Fernandes/UFPE (Coordenador de Workshops), Ítalo Cunha/UFMG (Coordenador de Palestras, Tutoriais e Mentoria), Artur Ziviani/LNCC (Coordenador de Painéis), Miguel Elias Mitre Campista/UFRJ (Coordenador de Minicursos), Leandro Villas/UNICAMP (Coordenador do Salão de Ferramentas), Daniel Fernandes Macedo/UFMG (Coordenador do Concurso de Teses e Dissertações), e Raquel Lopes/UFCG e Luis Carlos De Bona/UFPR (Coordenadores do Hackathon). Agradecemos também aos membros do nosso comitê local, que nos ajudaram na operacionalização de diversas tarefas relacionadas à organização: Avelino Zorzo/PUCRS, Carlos Raniery Paula dos Santos/UFSM, Cristiano Bonato Both/Unisinos, Guilherme Rodrigues/IFSUL Charqueadas, Jéferson Campos Nobre/UFRGS, Juliano Wickboldt/UFRGS, Marcelo Caggiani Luizelli/Unipampa, Marcelo da Silva Conterato/PUCRS, Rafael Esteves/IFRS, Rodrigo Mansilha/Unipampa, Tiago Ferreto/PUCRS, e Vinícius Guimarães/IFSUL Charqueadas. Agradecemos ao Luis Otávio Luz Soares, técnico administrativo do INF/UFRGS, pela imensa ajuda nos mais variados aspectos relacionados à organização do evento, assim como pelo apoio de Leandro Disconzi Vieira e Carlos Alberto da Silveira Junior, também técnicos administrativos da UFRGS. Por fim, agradecemos à nossa equipe local, formada por alunos de graduação, pós-graduação e pós-doutorando da UFRGS, pela dedicação e pronta ajuda sempre que precisamos: Arthur Selle Jacobs, Augusto Zanella Bardini, Bruno Dalmazo, Fernanda da Silva Bonetti, Guilherme Bueno de Oliveira, Guilherme Rotth Zibetti, Isadora Pedrini Possebon, Leonardo Lauryel, Libardo Andrey Quintero González, Lucas Bondan, Lucas Castanheira, Luciano Zembruzki, Mateus Saquetti, Jonatas Marques, Rafael Hengen Ribeiro, e Ricardo Parizotto.

Desejamos a todos um ótimo evento e uma semana produtiva e com muitas trocas de ideias em Gramado.

Alberto Egon Schaeffer Filho e Weverton Luis da Costa Cordeiro
Coordenadores Gerais do SBRC 2019

Mensagem do Coordenador de Workshops

Os Workshops do SBRC são uma importante oportunidade para aprofundamento no conhecimento de temas especializados ou emergentes na nossa comunidade científica. Como em anos anteriores, a coordenação manteve a *Chamada de Propostas de Workshops do SBRC*, que tem estimulado a comunidade brasileira de Redes e Sistemas Distribuídos a discutir a viabilidade de workshops com temas de pesquisa mais tradicionais, bem como estimular o debate para adoção de eventos que tratam de temas mais emergentes.

Tivemos uma diversidade salutar no Comitê de Avaliação em termos de localização geográfica e experiência, gerando, portanto, opiniões complementares sobre as propostas submetidas. Consideramos todas as propostas de altíssima qualidade, com temas, focos e escopo diversos. Nesta edição de 2019, buscamos equilibrar os benefícios trazidos aos participantes com a capacidade de alocação de espaços no evento. Desta forma, foram selecionados 10 Workshops de alta qualidade, além do tradicional Workshop da RNP (*WRNP*). Dentre as propostas aceitas, oito são reedições de workshops tradicionais do SBRC, a saber: **Gerência e Operação de Redes e Serviços** (*WGRS*), **Testes e Tolerância a Falhas** (*WTF*), **Pesquisa Experimental da Internet do Futuro** (*WPEIF*), **Segurança Cibernética em Dispositivos Conectados** (*WSCDC*), **Trabalhos de Iniciação Científica e Graduação** (*WTICG*), **Blockchain: Teoria, Tecnologias e Aplicações** (*WBlockchain*), **Clouds e Aplicações** (*WCGA*), e **Computação Urbana** (*CoUrb*). Como novidade, teremos dois novos workshops, a saber: o Workshop de **Teoria, Tecnologias e Aplicações de Slicing para Infraestruturas Softwarizadas** (*WSlice*) e o **Latin American Student Workshop on Data Communication Networks** (*LANCOMM Student Workshop*), que tem como público-alvo principal os alunos de pós-graduação e pesquisadores da América Latina, além de estreitar os relacionamentos da nossa comunidade com a Association for Computing Machinery (*ACM*).

Como coordenador dos Workshops do SBRC 2019, gostaria de agradecer a todos os envolvidos na seleção das propostas. Primeiramente aos coordenadores gerais do SBRC 2019, Weverton Cordeiro (UFRGS) e Alberto Egon Schaeffer Filho (UFRGS), pelo convite para a coordenação desta chamada de trabalhos, além de todo o apoio necessário para sua condução apropriada. Agradeço também a todos os membros do comitê de avaliação pelo imenso esforço nas revisões e discussões de alta qualidade de todas as propostas submetidas. Por fim, agradeço aos coordenadores dos workshops aceitos, pela dedicação no cumprimento dos prazos e na condução dos trabalhos internos de seleção dos artigos para seus respectivos eventos, mantendo a alta qualidade geral do SBRC. Os inscritos nos diversos workshops esperam ansiosamente pelas apresentações dos trabalhos e pelas frutíferas discussões que serão geradas.

Stênio Fernandes
Coordenador dos Workshops do SBRC 2019

Comitê de Programa

- Alex Borges (Universidade Federal de Juiz de Fora)
- André Grégio (UFPR)
- Billy Pinheiro (Universidade Federal do Pará)
- Carlos Ruggiero (University of Sao Paulo)
- Cesar Marcondes (Instituto Tecnológico da Aeronáutica)
- Diego Aranha (Aarhus University)
- Erick Lazaro Melo (UFSCar)
- Hermes Senger (Universidade Federal de São Carlos)
- Jean Martina (Universidade Federal de Santa Catarina)
- José Parente de Oliveira (Technological Institute of Aeronautics - ITA)
- Kalinka Castelo Branco (ICMC - USP)
- Lourenco Pereira Jr (Instituto Tecnológico de Aeronáutica - ITA)
- Luis Fernando Lopez (Universidade de Sao Paulo)
- Mário Alvim (Universidade Federal de Minas Gerais)
- Paulo Matias (Universidade Federal de São Carlos)
- Paulo Angelo Resende (Universidade de Brasília)
- Pedro Rosa (Universidade Federal de Uberlandia)
- Raimundo Jose de Araujo Macedo (UFBA)
- Ricardo Custódio (UFSC)
- Ricardo Sant'Ana (Instituto Militar de Engenharia)
- Rodolfo Meneguette (Instituto Federal de Educação, Ciência e Tecnologia de São Paulo)
- Rodolfo Villaca (Universidade Federal do Espirito Santo (UFES)
- Rodrigo Miani (Universidade Federal de Uberlândia)
- Rodrigo Rubira Branco (Intel)

Sumário

Sessão técnica 01 1

Geração de workload para treinamento, teste, avaliação e comparação de sistemas de detecção de intrusão baseado na replicação de características estatísticas 2
Moisés da S. Rodrigues (ITA), Sidnei Barbieri (ITA), José M. Parente de Oliveira (ITA), César A. C. Marcondes (ITA), Lourenço Alves Pereira (ITA)

Uma Proposta de Contramedida ao Ataque Jamming em Redes IEEE 802.15.4 utilizando Rádio Cognitivo 13
Fábio V. Almeida (Inatel), M. M. Bontempo (Inatel), J. R. Santos (Inatel), Antônio M. Alberti (Inatel)

HonIoT: Arquitetura de Honeynet com Controle de Propagação de Malwares para Dispositivos de IoT..... 24
Douglas B. de Godoy (UFSCar), Jaime F. Souza (UFSCar), Emerson R. A. Barea (IFTO), César A. C. Marcondes (ITA)

Sessão técnica 02 38

Uma avaliação de Algoritmos Criptográficos em Redes IEC 61850: Uma Abordagem Prática 39
Rafael B. Scarselli (UFF), Leonardo F. Soares (UFF), Igor M. Moraes (UFF)

O impacto das fragilidades da comunicação 802.11 em competições de robótica 53
Júlio C. M. de Resende (UFSJ), André F. J. Santos (UFSJ), Thiago S. Gomides (UFSJ), Massilon Lourenço (UFSJ), Flávio L. Schiavoni (UFSJ), Marcos A. M. Laia (UFSJ)

Breve Comparação de Ferramentas para Análise Estática de Código Malicioso 58
William Akihiro Alves Aisawa (IFRN), Pedro Henrique Lara Campos (UFRN), Cesar Augusto Cavalheiro Marcondes (UFRN), Paulo Matias (UFRN)

Sessão técnica 03 63

Modelo de maturidade de segurança cibernética para os órgãos da Administração Pública Federal 64
Antonio João Gonçalves de Azambuja (AGU), João Souza Neto (UCB)

Coleção de dados sobre ataques a dispositivos de Internet das Coisas 78
Marcos Felipe B. de Abreu (UFG), Thierson Couto Rosa (UFG), Kleber Vieira Cardoso (UFG)

Uma Taxonomia para Descrever e Caracterizar Estratégias de Mitigação de Ataques DDoS em Ambientes IoT Usando Tecnologias SDN 91
Esau Silva, Felipe S. Dantas Silva, Marcilio O. O. Lemos, Augusto Venâncio Neto

II Workshop de Segurança Cibernética em Dispositivos Conectados
SBRC 2019
Sessão Técnica 1

Geração de *workload* para treinamento, teste, avaliação e comparação de sistemas de detecção de intrusão baseado na replicação de características estatísticas

Moisés da S. Rodrigues, Sidnei Barbieri, José M. Parente de Oliveira
César A. C. Marcondes, Lourenço Alves Pereira

[1]Divisão de Ciência da Computação
Instituto Tecnológico de Aeronáutica—ITA

{moises,sidnei,parente,cmarcondes,ljr}@ita.br

Resumo. *Este artigo apresenta o resultado de um trabalho de geração de datasets para sistemas de detecção de intrusão. Este processo utiliza datasets existentes constituídos por pacotes, assim como suas características estatísticas. A modelagem da distribuição de variáveis como, por exemplo, o número de pacotes contíguos de mesmo rótulo, é responsável pela geração de novas estruturas, as quais preservam características estatísticas semelhantes. Como resultado, obtivemos uma nova base de dados, que pode ser utilizada não apenas para o treinamento e teste de novos sistemas de detecção, mas também para a reavaliação de outros sistemas previamente desenvolvidos. Isto possibilita a comparação com os resultados de novos trabalhos.*

Abstract. *This paper presents the results of a dataset generation work for intrusion detection systems. This process uses existing datasets made up of packets, as well as their statistical characteristics. The modeling of the distribution of variables, such as the number of contiguous packets of same label, is responsible for the generation of new structures, which preserve similar statistical characteristics. As a result, we have obtained a new database, which can be used not only for the training and testing of new detection systems, but also for the re-evaluation of other previously developed systems. This enables comparison with the results of new jobs.*

1. Introdução

Considerando a relevância de sistemas de detecção de intrusão (IDS), tanto os baseados em uso indevido quanto os baseados em anomalias, para a segurança cibernética, verifica-se a necessidade da disponibilidade de bases de dados confiáveis, compostas por tráfego normal (benigno), e também por tráfego de ataque (maligno), as quais são denominadas *datasets* [Lippmann et al. 2000]. O principal problema, contudo, reside na identificação de bases disponíveis, uma vez que tal disponibilização implica em considerações relacionadas à segurança da informação.

Partindo da premissa de que os IDS, em sua finalidade principal, destinam-se à detecção de tráfego real, vislumbramos que o ideal seria que os *datasets* fossem os mais realistas possível, representando ambientes operacionais de larga escala [Sommer and Paxson 2010]. Sob essa ótica, verificamos que os *datasets* deveriam ser construídos

exclusivamente a partir de tráfego real coletado diretamente em ambientes de redes em produção. Infelizmente tal abordagem tem como principal óbice a dificuldade de obtenção desses dados, uma vez que não é comum o compartilhamento de dados reais, o que também afeta, ainda que parcialmente, os critérios da completude das capturas.

Ao considerarmos *datasets* baseados em tráfego real, devemos considerar que além dos dados desejados para a pesquisa, há, também, informações relevantes para as organizações, como informações sobre sua estrutura de rede ou, ainda, sobre o negócio. Desse modo, não é comum que dados sejam efetivamente compartilhados, o que se justifica pelo argumento da garantia da privacidade. Convém, ainda, ressaltar que, mesmo nas situações em que o acesso é franqueado, muitas vezes tal acesso e a utilização está condicionada à aprovação dos custodiantes, mediante o preenchimento de formulários com a assinatura de concordância dos termos de uso aceitável [Nehinbe 2011], o que, eventualmente, pode impactar na celeridade da pesquisa. Há, ainda, situações em que os dados, antes de serem disponibilizados, são submetidos a um processo anonimização, o que, por vezes, chega a desfigurar a informação, tornando-a inutilizável.

Em virtude das dificuldades inerentes à obtenção de *datasets* baseados na coleta de tráfego real, é fato que existe uma carência de *datasets* que sejam atuais, contenham representação de tráfego real e que seja coletado em redes modernas [Zuech et al. 2015]. Em consequência, verificamos que a alternativa que se apresenta mais viável consiste na geração de tráfego simulado para a composição de *datasets*, definidos como *benchmamarks datasets* [Rao and Naidu 2017]. Na verdade, tal conclusão não é recente, [Scott and Wilkins 1999] já apontavam a necessidade da geração de *datasets* artificiais, sujeitos, contudo, à manutenção de determinadas características de *datasets* reais, como os tipos de padrão identificados no tráfego real.

Gerar *datasets* artificiais, todavia, não é uma tarefa simples, uma vez que deve-se garantir que eles sejam completos, válidos e adequados [Sharafaldin et al. 2017]. Deve-se, também, considerar que, à medida que ataques mais sofisticados são desenvolvidos, algumas atividades maliciosas podem se tornar obsoletas, sendo, portanto, desejável que os datasets sejam dinâmicos, de maneira que possam ser modificáveis, extensíveis e reprodutíveis. Ainda quanto ao tamanho, deve-se ter um cuidado especial para que as bases tenham um tamanho tal que possibilite sua utilização. A quantidade excessiva de registros pode inviabilizar a a utilização do *datasets* como um todo, implicando na necessidade da extração de amostras que, por mais que decorram da aplicação de princípios estatísticos, aumentam a probabilidade de descartar tráfego relevante, tanto sob o ponto de vista de protocolos quanto de ataques [Tavallaee et al. 2009].

A geração de *datasets* artificiais deve garantir que eles sejam completos, válidos e adequados [Sharafaldin et al. 2017]. Na medida que ataques mais sofisticados são desenvolvidos, algumas atividades maliciosas podem se tornar obsoletas, sendo desejável que os datasets sejam dinâmicos, adaptáveis, extensíveis e reprodutíveis. O tamanho das bases de dados também é significativo. A quantidade excessiva de registros pode inviabilizar a utilização do *datasets*, exigindo a extração de amostras que podem implicar na perda de tráfego relevante, tanto sob o ponto de vista de protocolos quanto de espécies de ataque [Tavallaee et al. 2009].

Diversos trabalhos sobre técnicas de avaliação de IDS e de detecção de intrusão vêm

sendo desenvolvidos nos últimos anos, muitos deles baseados em *datasets* distintos, o que dificulta sua comparação com os demais trabalhos desenvolvidos com a mesma finalidade. Assim, incentivado pela carência de *datasets* atuais, contendo tráfego real e coletado em redes modernas [Zuech et al. 2015], o presente trabalho visa propor um método para geração de tráfego simulado, a partir do qual são constituídos *datasets* aplicáveis, entre outros aspectos, na avaliação comparativa de IDS.

Em face do exposto, este artigo propõem e demonstra a aplicação de um método para a geração artificial de *workload* aplicável ao treinamento, teste, avaliação e comparação de sistemas de detecção de intrusão, baseado na replicação de características estatísticas. A seguir, na Seção 2 são discutidos trabalhos relacionados. Já a Seção 3 é dedicada à descrição do método desenvolvido, enquanto que a Seção 4 detalha a sua aplicação no estudo de caso proposto. Por fim, na Seção 5 são apresentadas as conclusões, as limitações e a proposta de trabalhos futuros.

2. Trabalhos relacionados

A geração de *datasets* para sistemas de detecção de intrusão tem sido objeto de trabalhos há mais de 20 anos, sendo, segundo [McHugh 2000], o DARPA 98/99 o primeiro digno de avaliação. Desde então, outros trabalhos propuseram a avaliação ou a geração de novos conjuntos de dados, visando sempre à garantia de bases consideradas realistas e válidas. Em que pese a antiguidade dessa base bem como do KDD Cup 99, nela baseado, esses *datasets* vêm sendo utilizados em diversas pesquisas. O principal impedimento a eles relacionado está na sua desatualização, uma vez que não refletem a evolução do tráfego nem dos ataques. [Tavallaee et al. 2009] propuseram uma análise de problemas estatísticos relacionados ao KDD 99 gerando um novo conjunto de dados, o NSL-KDD, consistindo em uma otimização do conjunto original. Apesar da melhoria alcançada, por ter como base dados já obsoletos, o *dataset* resultante também padeceu das mesmas limitações.

Tendo como referência a busca pelo realismo, [Shiravi et al. 2012] apresentaram 5 prerrequisitos preliminares a serem considerados quando da geração de um *dataset*, a saber: conter tráfego de rede realista, ser rotulado, conter interações entre todas as redes, conter capturas de dados completas e contemplar múltiplos cenários de ataque. Em consequência, seu trabalho culminou na geração de *datasets* baseados na descrição de cenários de ataque e na distribuição matemática do comportamento das entidades presentes em redes computacionais. A despeito dos bons resultados obtidos, conforme destacado pelos próprios autores tal abordagem se limita a contextos particulares, com requisitos e aplicações específicos.

[Gharib et al. 2016] apresentaram um *framework* para avaliação de *datasets* composto por 11 características que as bases deveriam apresentar para que fosse considerados válidas, a saber: diversidade de ataques, anonimato, protocolos disponíveis, completude da captura, completude da interação, completude da configuração de rede, completude do tráfego, conjunto de *features*, heterogeneidade, rotulação e metadados. Adicionalmente, realizaram uma comparação entre 5 abordagens para obtenção de *datasets*, com suas vantagens e desvantagens: *replay* de tráfego de *datasets* públicos, geração artificial de tráfego, desenvolvimento de estratégias de arquitetura para geração de tráfego, simulação e representação física de uma rede real. Com base em seu *framework*, eles avaliaram 11 *datasets*, com *scores* que variam entre 0 e 1, porém não geraram nenhum *dataset* capaz de

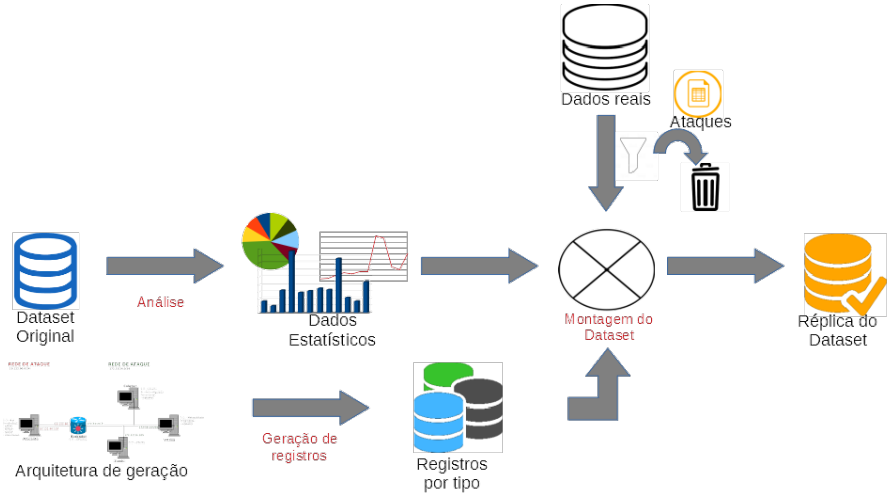


Figura 1. Estrutura interna de um *dataset*.

se adequar ao *framework* proposto.

Na tentativa de garantir a validade e o realismo de um *dataset* artificial [Haider et al. 2017] desenvolveram uma abordagem para avaliação de *datasets* mediante uma modelagem baseada em lógica *fuzzy*, o que lhe permitiu comparar alguns *datasets* em abordagem semi-quantitativa e gerar um novo conjunto de dados, o NGIDS-DS. Com intenções semelhantes, tendo por base os onze critérios de [Gharib et al. 2016], [Sharafaldin et al. 2017] geraram uma nova base de dados valendo-se da abstração do comportamento das interações humanas e simulação de cenários de ataque, o CIC-IDS. Muito embora tenham obtido bons resultados, tais abordagens não viabilizam a comparação de trabalhos voltados ao desenvolvimento de IDS obtidos anteriormente, dificultando a percepção dos avanços alcançados.

Embora existam diversos métodos para geração de *datasets*, a justa comparação entre sistemas de detecção de intrusão antigos e contemporâneos é prejudicada pela evolução do tráfego das redes de computadores. Assim, o presente trabalho oportuniza tanto a geração de *datasets* atuais quanto a extração de subconjuntos com características comparáveis às de *datasets* utilizados em avaliações antigas.

3. Método para Geração de *Workload*

Em linhas gerais, em nosso método focamos em características de datasets baseados em pacotes que pudessem ser reproduzidas, sob o ponto de vista estatístico. Cientes, contudo, das dificuldades associadas à análise temporal, optamos por não realizar tal análise. Assim, de maneira resumida, o método foi desenvolvido de acordo com as seguintes etapas, que serão abordadas nas próximas subseções: análise do *dataset*, visando à identificação de características estatísticas reprodutíveis, seguida da geração do esqueleto do novo *dataset*; obtenção de pacotes para preenchimento do esqueleto, realizada mediante a geração de tráfego maligno e a coleta de tráfego benigno; e montagem do novo *dataset*. A Figura 1 ilustra as etapas realizadas.

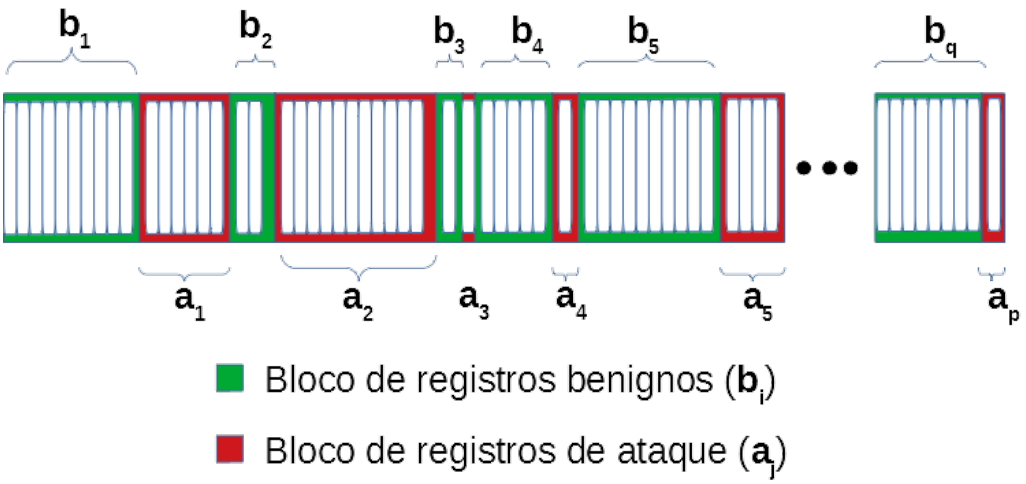


Figura 2. Estrutura interna de um *dataset*

3.1. Análise Estatística do *Dataset*

Ao olhar para um *dataset* baseado em pacotes, observamos que sua estrutura se apresenta em blocos, cada bloco com k registros com mesmo rótulo (benigno/maligno), $k \geq 1$. Assim a cada vez que um registro de um tipo distinto do anterior surge, conforme a sequência de pacotes no *dataset*, considerado o rótulo, inicia-se um novo bloco, conforme podemos ver na Figura 2. Tal constatação nos permitiu identificar como possível variável a ser analisada o tamanho de cada bloco dentro do universo de blocos com pacotes com mesmo rótulo, ou seja, o número de registros consecutivos com mesmo rótulo até que surja um registro com rótulo distinto. Identificamos, ainda, outras possíveis características passíveis de análise, como os percentuais correspondentes aos pacotes de cada serviço ou mesmo os associados aos diferentes protocolos nas diferentes camadas. Verificamos, entretanto, que todas a adoção dessas outras características para a finalidade de referência para a replicação demandaria uma análise adicional vinculada a outras características, o que, neste trabalho, nos levou a focar apenas no tamanho do bloco.

Considerando dois conjuntos distintos de blocos, a saber blocos compostos por pacotes benignos e blocos compostos por pacotes malignos, a análise em cada conjunto da variável tamanho do bloco nos levou a uma distribuição de frequências, cuja submissão ao processo de *fitting*, conforme o gráfico de Cullen e Frey, nos permitiu enquadrá-la em uma função de distribuição de probabilidade, a qual serviu de base para replicação. O ambiente R, utilizado por nós para essa finalidade, apresenta funções adequadas a tal tarefa, como exemplifica a Figura 3. Assim, uma vez definida a função para replicação da distribuição de frequência das variáveis tamanho do bloco com pacotes benignos e tamanho do bloco com pacotes malignos, passamos a montar um esqueleto para a recepção de novos pacotes, ou seja, definimos os tamanhos de cada bloco, sem populá-los com novos pacotes. Para executar tal tarefa, foi necessário definirmos o número de blocos a ser gerado $2h$ (h blocos com registros benignos e h blocos com registros malignos, o que foi feito em função do número n de registros (pacotes) do *dataset* original e das médias das funções de distribuição, μ_a no caso do tamanho dos blocos de ataque e μ_b no caso do tamanho dos blocos benignos, conforme a Equação 1.

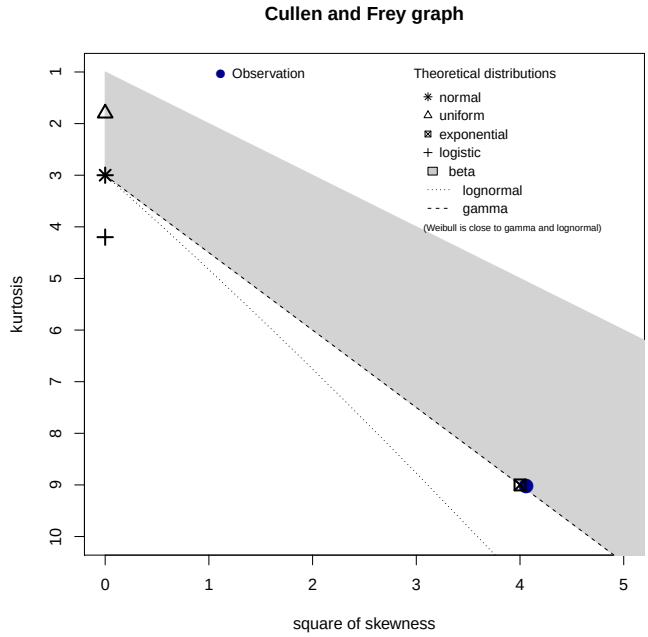


Figura 3. Seleção no R da distribuição associada ao tamanho dos blocos benignos com base na assimetria (*skewness*) e curtose (*Kurtosis*). (Fonte: Autor)

$$h = \frac{n}{\mu_b + \mu_a} \tag{1}$$

Uma vez montado o esqueleto do novo *dataset*, passamos, então à obtenção de novos pacotes para populá-lo.

3.2. Obtenção de pacotes e Montagem do *dataset*

Para obter os dados para preencher o esqueleto criado conforme a Seção 3.1, dividimos nosso trabalho em duas etapas: geração de tráfego malicioso e coleta de tráfego benigno. A geração do tráfego malicioso, considerando a intenção de replicar os tipos de ataques sem focar nos aspectos temporais, se baseou em um *testbed* simples, composto por duas redes, uma denominada rede de ataque e a outra denominada rede de defesa, ambas segregadas entre si por um roteador baseado no sistema operacional FreeBSD, conforme a Figura 4. Para gerar os ataques, utilizamos o sistema operacional Kali Linux, criado especificamente com a finalidade de realizar testes de penetração. Para o papel de *host* vítima, selecionamos o sistema Metasploitable 2, concebido a partir do sistema operacional Ubuntu, com vulnerabilidades conhecidas e intencionais. Inserimos, também *hosts* denominados zumbis, os quais poderiam ser utilizados pelos atacantes como pontes para a realização de ataques e, por fim, coletamos o tráfego gerado por meio de um *host* com sua placa de rede configurada em modo promíscuo, com o sistema Ubuntu e a ferramenta TShark, vinculada à ferramenta de captura de pacotes Wireshark.

Com relação ao tráfego de benigno, cientes de que sua geração não costuma ser intencionalmente planejada, o que dificulta sua replicação em ambiente de laboratório, adotamos como linha de ação a busca de dados em fontes reais com nível de confiabilidade

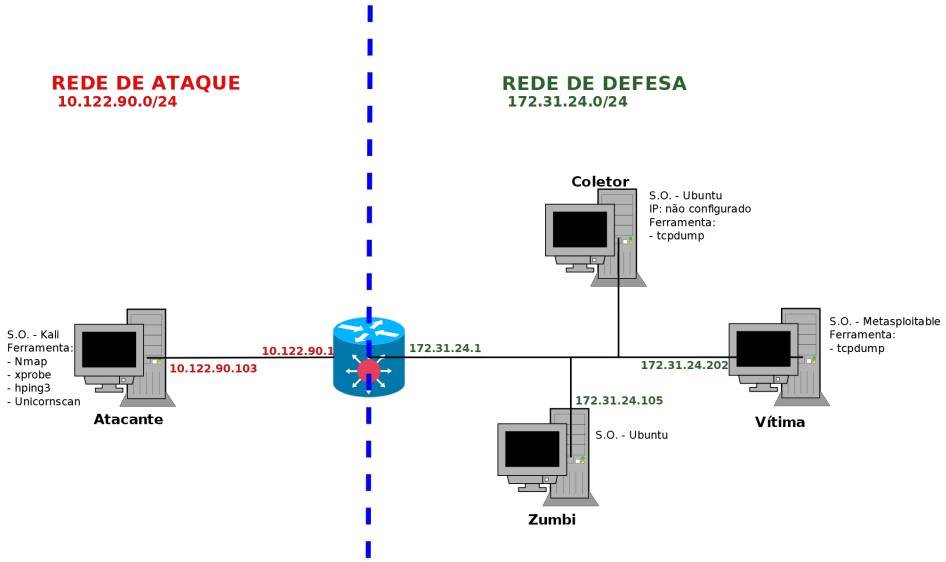


Figura 4. *Testbed* para geração dos ataques do *dataset*.

julgado aceitável. Assim, para complementar o tráfego malicioso gerado, nos valemos dos dados disponibilizados pelo Mawilab [Fontugne et al. 2010] composto por pacotes de múltiplos protocolos e serviços. De uma forma simplificada, o Mawilab disponibiliza dados reais coletados em um *link* transpacífico, acrescidos de um arquivo com os resultados de heurísticas de detecção aplicadas que possibilitam a identificação dos ataques presentes nos dados originais. Assim, utilizando os arquivos que identificam os ataques como filtro, conseguimos obter tráfego genérico, benigno e, consideravelmente, confiável quanto à ausência de possíveis ataques.

Por fim, a última etapa realizada foi a população do esqueleto do *dataset*, preenchendo cada um dos blocos, tanto os de dados malignos quanto os de benignos, respectivamente, com os pacotes maliciosos gerados conforme o *testbed* e os pacotes benignos filtrados a partir daqueles coletados do Mawilab, os quais estão dispostos em uma estrutura de fila. Tal composição garante que, ao final do processo, obtenhamos um novo conjunto de dados, com pacotes completamente distintos dos originais, mantendo, contudo a mesma característica estatística relativa à quantidade de pacotes benignos e malignos contíguos.

4. Estudo de Caso

Com o intuito de demonstrar na prática a aplicação do método de geração de *datasets*, selecionamos o *dataset* desenvolvido no trabalho de [Barbieri 2019], voltado à avaliação do método determinístico por ele proposto para detecção de ataques de TCP *scanning*. Por ser baseado em pacotes e não em fluxos e por sua simplicidade, contendo ataques de um único tipo, julgamos adequado utilizar esse *dataset* como entrada para o nosso método. Segundo [Barbieri 2019], a motivação para a geração de uma nova base de dados foi a ausência de *datasets* que contivessem ataques específicos e identificados de TCP *scanning*, responsáveis por $2,1\,\%$ dos pacotes do tráfego global anual, que, até o final de 2020, pode chegar ao volume anual de $2,3$ Zettabyte [Glatz and Dimitropoulos 2012, Idzikowski et al. 2018, Lee and Lee 2012]. Quanto à composição, o *dataset* apresentado por Barbieri continha $1.603.761$ registros, sendo $1.239.293$ pacotes benignos

Tabela 1. Comparação entre os *datasets* de Barbieri e nossa réplica

Característica	Barbieri	Réplica	Diferença
Total de registros	1.603.761	1603761	—
registros de ataque	364.468	334.660	$-8,18\%$
registros benignos	1.239.293	1.269.101	$2,41\%$
blocos de ataque	281.662	268.637	$4,98\%$
blocos benignos	281.662	268.637	$4,98\%$

e 364.468 pacotes maliciosos. O principal ponto a se destacar no conjunto original apresentado é a forma como os ataques foram obtidos, a saber, mediante a utilização de múltiplas técnicas e *scripts* disponíveis na ferramenta Nmap.

Considerando o conjunto original de dados, identificamos 281.663 blocos com registros normais ou benignos e a mesma quantidade de blocos com pacotes de ataque ou malignos. Submetendo as distribuições de frequência das variáveis tamanho dos blocos ao processo de *fitting*, primeira etapa de nosso método de geração, nos 281.663 blocos com pacotes malignos identificamos uma função Beta, enquanto para a variável tamanho dos blocos relativos ao tráfego normal, com valores coletados nos 281.663 blocos benignos identificamos uma função Gamma, mais especificamente uma função Exponencial. As médias das funções de distribuição identificadas foram, respectivamente, $\mu_a = 1,57$ e $\mu_b = 4,40$, de maneira que, por meio da Equação 1, apresentada na seção 3, aproximadamente, definimos o número de blocos $2h = 537.274$, correspondente a 268.637 blocos com pacotes benignos e 268.637 blocos com pacotes referentes a ataques de TCP *scanning*.

A geração de ataques foi realizada segundo o *testbed* apresentado na Figura 4 utilizando as seguintes ferramentas: Nmap, Angry IP Scan, Dmitry, Hping3, Masscan, Netcat e Unicornscan. No caso do Nmap, foram utilizadas as seguintes técnicas: *ack scan*, *connect scan*, *custom scan*, *decoy scan*, *fin scan*, *fragmented connect scan*, *maimon scan*, *protocol scan*, *stealth syn scan*, *windows scan* e *xmas scan*. Ainda nessa etapa, o tráfego malicioso foi complementado pelo tráfego benigno coletado a partir do Mawilab submetido ao filtro para expurgar os ataques, conforme já explicado na subseção 3.2. Os pacotes de tráfego normal e de ataque obtidos foram utilizados para mobiliar os blocos, conforme a última etapa, chegando a um conjunto final com $1.269.101$ registros benignos e 334.660 registros de ataque do tipo TCP *scanning*. A Tabela 1 compara características gerais do *dataset* original e do *dataset* que geramos, denotando a semalhança entre eles. Complementarmente, aplicamos os critérios de [Gharib et al. 2016] para avaliação da validade de um *dataset*, tendo obtido para ambos os *datasets* o score $0,68$ o que reforça a similaridade entre eles, desta vez sob o ponto de vista da validade.

Repetindo os experimentos realizados no trabalho de [Barbieri 2019], cuja comparação está disposta na Tabela 2, a despeito da similaridade estatística entre a distribuição do tamanho dos blocos, observamos uma forte queda no desempenho do método de detecção proposto, com o *True Positive Rate* passando de 89% para 55%. Tal discrepância, em nossa análise, pode ser um indício de que, talvez, os resultados originais tenham sido enviesados em consequência da ferramenta utilizada, dado que essa foi a principal alteração nos dois conjuntos de dados utilizados para avaliação, sendo o método de detecção baseado em

pacotes e *stateless*.

Tabela 2. Comparação do desempenho do método de detecção proposto por Barbieri frente ao *dataset* original e o que geramos

Dataset utilizado	CC[1] (%)	TPR[2] (%)	TNR[3] (%)
Dataset original	97,59%	89,39%	100,00%
Nossa réplica	82,11%	55,24%	100,00%

[1]CC: Acurácia [2]TPR: *True Positive Rate*. [3]TNR: *True Negative Rate*

Apesar da diferença dos resultados obtidos na avaliação do método de detecção, verificamos utilidade da geração de uma nova base de dados semelhante à original, uma vez que viabiliza a reavaliação e a comparação com novos trabalhos.

5. Conclusões e Trabalhos Futuros

Sistemas de detecção desenvolvidos ou adquiridos não podem ter sua eficácia garantia senão pela submissão a testes e avaliações, o que pressupõe a existência de bases de dados capazes de oferecer o desafio necessário à avaliação. Assim, o presente trabalho teve como objetivo principal desenvolver um método capaz de gerar novos *datasets* a partir da identificação e replicação de características estatísticas presentes em *datasets* já existentes.

A replicação do *dataset* de Barbieri possibilitou verificar que é possível reproduzir as características de um *dataset* para gerar um conjunto de dados distinto, mas que preserva semelhanças estatísticas. Desta forma, pesquisas que poderiam ter a avaliação de seu desempenho prejudicado em virtude da utilização de *datasets* obsoletos, podem obter resultados mais realistas e significativos, comparáveis aos resultados de outras pesquisas.

Além disso, a diversidade conseguida com a construção de novos *datasets* favorece a avaliação de sistemas de detecção de intrusão. O experimento realizado demonstrou que a replicação de um *dataset*, apesar de mantidas algumas de suas propriedades estatísticas, pode alterar fortemente os resultados da avaliação de sistemas de detecção pela inserção de novos tipos de tráfego ou pela exclusão de tipos presentes no *dataset* de origem. Tal constatação aponta para uma aplicação adicional não prevista da replicação, a saber, a possível correção de eventuais falhas na avaliação de um *dataset* gerado artificialmente e com limitação no emprego de ferramentas de ataque.

Por outro lado, a grande diversidade de características existente entre os *datasets* identificados é desfavorável à extração de características comuns e, consequentemente, à sua generalização. Alguns trabalhos relacionados apresentam critérios bem definidos para a avaliação de *datasets*, os quais foram úteis para a identificação das caraterísticas mais significativas para o processo de generalização. Com efeito, o método proposto ficou restrito à reprodução de *datasets*. Não há qualquer impedimento à sua utilização para a extensão de bases já existentes, acrescentando novos dados ou mesmo alterando algumas das características estatísticas identificadas. Tal trabalho, visando à criação de novas bases em contextos específicos, pode ser objeto de novas pesquisas.

Como trabalhos futuros, pretendemos realizar a análise de outros *datasets* que contemplam mais tipos de ataques. Outro ponto a ser abordado é a construção de *testbed* realístico, que permita experimentação com configuração mais flexível, sendo possível

gerar tráfego benigno sintético e mesclá-lo com o tráfego maligno. Uma especialização deste método pode ser focada em Internet das Coisas—IoT. Ademais, a análise temporal também será objeto de futuros trabalhos.

Agradecimentos

Agradecemos Exército Brasileiro e ao Comando de Defesa Cibernética, ITA/PPG em Aplicações Operacionais pelo apoio financeiro.

Referências

[Barbieri 2019] Barbieri, S. (2019). Método para a detecção de pacotes produzidos por *scanning* tcp. Mestrado em engenharia eletrônica e computação, Instituto Tecnológico de Aeronáutica, São José dos Campos, SP.

[Fontugne et al. 2010] Fontugne, R., Borgnat, P., Abry, P., and Fukuda, K. (2010). Mawilab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 8:1–8:12, New York, NY, USA. ACM.

[Gharib et al. 2016] Gharib, A., Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2016). An evaluation framework for intrusion detection dataset. In *2016 International Conference on Information Science and Security (ICISS)*, pages 1–6.

[Glatz and Dimitropoulos 2012] Glatz, E. and Dimitropoulos, X. (2012). Classifying internet one-way traffic. In *Proceedings of the 2012 Internet Measurement Conference*, IMC'12.

[Haider et al. 2017] Haider, W., Hu, J., Slay, J., Turnbull, B., and Xie, Y. (2017). Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *J. Netw. Comput. Appl.*, 87(C):185–192.

[Idzikowski et al. 2018] Idzikowski, F., Chiaraviglio, L., Liu, W., and van de Beek, J. (2018). Future internet architectures and sustainability: An overview. In *2018 IEEE International Conference on Environmental Engineering (EE)*, pages 1–5.

[Lee and Lee 2012] Lee, Y. and Lee, Y. (2012). Toward scalable internet traffic measurement and analysis with hadoop. *SIGCOMM Comput. Commun. Rev.*, 43(1):5–13.

[Lippmann et al. 2000] Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K. (2000). The 1999 darpa off-line intrusion detection evaluation. *Comput. Netw.*, 34(4):579–595.

[McHugh 2000] McHugh, J. (2000). Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Inf. Syst. Secur.*, 3(4):262–294.

[Nehinbe 2011] Nehinbe, J. O. (2011). A critical evaluation of datasets for investigating idss and ipss researches. In *2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS)*, pages 92–97.

[Rao and Naidu 2017] Rao, C. M. and Naidu, M. M. (2017). A model for generating synthetic network flows and accuracy index for evaluation of anomaly network intrusion detection systems. *Indian Journal of Science and Technology*, 10(14).

[Scott and Wilkins 1999] Scott, P. D. and Wilkins, E. (1999). Evaluating data mining procedures: techniques for generating artificial data sets. *Information & Software Technology*, 41(9):579–587.

[Sharafaldin et al. 2017] Sharafaldin, I., Gharib, A., Lashkari, A. H., and Ghorbani, A. A. (2017). Towards a reliable intrusion detection benchmark dataset. *Software Networking*, 2017(1):177–200.

[Shiravi et al. 2012] Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.*, 31(3):357–374.

[Sommer and Paxson 2010] Sommer, R. and Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316.

[Tavallaee et al. 2009] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6.

[Zuech et al. 2015] Zuech, R., Khoshgoftaar, T. M., Seliya, N., Najafabadi, M. M., and Kemp, C. (2015). A new intrusion detection benchmarking system. In *The Twenty-Eighth International Flairs Conference*.

Uma Proposta de Contramedida ao Ataque Jamming em Redes IEEE 802.15.4 utilizando Rádio Cognitivo

Fábio V. Almeida [1], **M. M. Bontempo** [1],
J. R. Santos [1], **Antônio M. Alberti**[1]

[1]ICT Lab - Instituto Nacional de Telecomunicações (Inatel),
CEP 37540-000 - Santa Rita do Sapucaí, Minas Gerais, Brasil.

fabio.almeida@mtel.inatel.br, mariliamartins@gee.inatel.br,

joserodrigo@gec.inatel.br, alberti@inatel.br

Abstract. *Wireless sensor networks use shared media for data transmission, making it easy for an attacker to interfere with these networks via a denial of service attack known as Jamming. These attacks can be done through equipment known as Jammers, which emit random data signals, in order to overturn communications from sensor networks. In this paper, we study the characteristics of the jamming attack in IEEE 802.15.4 networks and propose a solution to this interference attack using a software-defined cognitive radio system to detect and protect the network.*

Resumo. *As redes de sensores sem fio utilizam de meios de comunicação compartilhados para a transmissão de dados. Com isto, se torna fácil para um atacante interferir nas transmissões destas redes através de ataques de negação de serviço conhecidos como Jamming. Estes ataques podem ser feitos através de dispositivos chamados de Jammers, que emitem sinais aleatórios de dados, visando derrubar as comunicações das redes de sensores. Neste artigo, estudamos as características do ataque de interferência Jamming sobre a rede IEEE 802.15.4 e propomos uma contramedida a esse ataque utilizando um sistema de rádio cognitivo definido por software para detectar e proteger a rede.*

1. Introdução

As Redes de Sensores sem Fio (RSSF) consistem em centenas ou até mesmo milhares de dispositivos pequenos, cada um com capacidade de detecção, processamento e comunicação para monitorar o ambiente físico. As RSSF operam através de um meio de comunicação compartilhado, se tornando assim alvos para ataques de negação de serviço, uma vez que um usuário mal intencionado pode reduzir a disponibilidade do meio [Torrieri 1985]. Um destes ataques se caracteriza como sendo o envio de um sinal em um meio de comunicação compartilhado com o intuído de reduzir sua disponibilidade (através da redução da relação sinal ruído do meio). Este ataque é conhecido como *Jamming* [Wood et al. 2007].

Jamming é definido como sendo um tipo especial de ataque de negação de serviço (em inglês, DOS - *Denial Of Service*). Wood e Stankovic definem DOS como sendo "qualquer evento que diminua ou elimine a capacidade de uma rede para executar sua

função esperada" [Wood and Stankovic 2002]. Portanto, se torna necessário o uso de sistemas de segurança que protejam as RSSF contra estas interferências propositais, principalmente quando levamos em consideração as limitações deste tipo de rede, como baixa capacidade computacional, memória limitada e baixa eficiência energética.

O objetivo deste artigo é descrever o ataque de negação *Jamming* e seus efeitos sobre uma rede de sensores IEEE 802.15.4, assim como, propor uma solução a este tipo de ataque através do uso de rádio cognitivo. Um rádio cognitivo pode analisar as frequências de transmissão e recepção de um meio compartilhado e alterar suas características [Mitola and Maguire 1999]. Para testarmos a ferramenta de *jamming*, foi desenvolvido um agente para a ferramenta de gerência Zabbix com a função de monitorar o canal de transmissão dos sensores e sua mudança. Ainda, realizamos a criação de uma rede de sensores utilizando um roteador de borda SmartRf06EBK™ e sensores CC2650-sensortag™ fabricados pela Texas Instruments para avaliar em campo a contramedida proposta.

Este artigo está organizado da seguinte forma: na Seção II estão descritas as características do padrão do *Institute of Electrical and Electronic Engineers* IEEE 802.15.4. Na Seção III estão descritas as características e os tipos de ataque de *Jamming* conhecidos. Na Seção IV apresenta-se uma proposta de arquitetura de Rádio Cognitivo que emprega um sistema de Rádio definido por Software como contramedida ao ataque Jamming. Na Seção V listamos os trabalhos relacionados. Na seção VI é demonstrado os resultados experimentais e a análise da ferramenta. Na seção VII concluimos o artigo.

2. Característica do Padrão IEEE 802.15.4

O IEEE 802.15.4 é um padrão para *Wireless Personal Area Network* (WPAN) que propõe uma rede de dados pessoal sem fio de baixa taxa de dados. É caracterizada por permitir uma rede de baixa complexidade e um tempo de vida maior de bateria. Suas potenciais aplicações são sensores, brinquedos interativos, crachás inteligentes, controles remotos [sta 2015]. Seu alcance varia de 10 a 100 metros, com uma potência máxima de saída geralmente de 0 dBm. O padrão define as características da camada física (PHY) e *Medium Access Control* (MAC) do modelo de referência *Open System Interconnection* (OSI).

Conforme a Tabela 1, a camada física do padrão IEEE 802.15.4 opera em três faixas de frequência livres, sendo elas: 868MHz, 915MHz e 2,4GHz *Industrial, Scientific and Medical* (ISM). Assim como as frequências de 314–316MHz, 430–434MHz, e 779–787MHz para *Low Rate - Wireless Personal Area Network* (LR-WPAN) na China e 950–956MHz no Japão [sta 2015].

Conforme a Figura 1, o padrão divide o espectro em um total de 27 canais, sendo 1 canal para a frequência de 868MHz, 10 canais para a frequência de 915MHz e 16 canais para a frequência de 2,4GHz.

Direct Sequence Spread Spectrum (DSSS) é utilizado nas frequências de 868/915 MHz junto com a modulação *Binary Phase Shift Keying* (BPSK), que permite taxas de transmissão até 20 Kbps e até 40 Kbps. Já na frequência de 2.4 GHz utiliza-se modulação *Quadrature Phase Shift Keying* (QPSK), que permite uma taxa de transmissão até 250 kbps [sta 2015] [Misic and Misic 2008].

A partir da revisão IEEE 802.15.4a foram introduzidas novas opções para a ca-

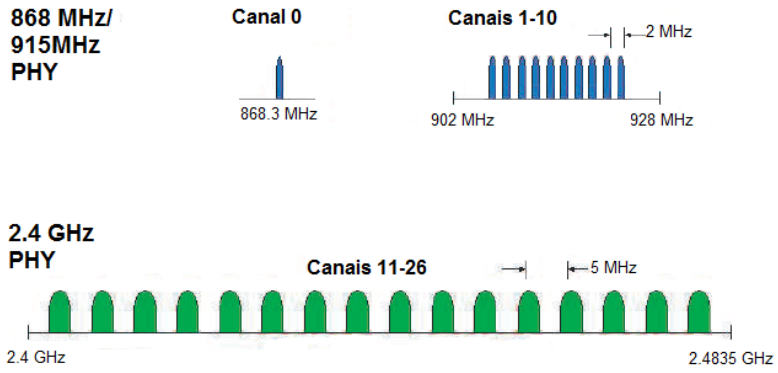


Figura 1. Espectro de radiofrequência IEEE 802.15.4 [Ergen 2004].

PHY	Banda de Frequência (MHz)	Taxa de Espalhamento (Kchip/s)	Modulação	Taxa de Bits (Kb/s)	Taxa de Símbolos	Símbolos
869/915	868-868,6	300	BPSK	20	20	BINÁRIO
	902-928	600	BPSK	40	40	BINÁRIO
2.450	2400-2483,5	2000	O-QPSK	250	62,5	16 símbolos

Tabela 1. Banda de Frequência e Taxa de Dados [Misic and Misic 2008].

mada física, permitindo um aumento nas taxas de dados e capacidade de alcance preciso e novas aplicações baseadas em informações sobre distância e posições dos dispositivos na rede. Estas melhorias começaram a partir da versão lançada em 2006 que já permitia se alcançar taxas de 250Kbps para as frequências de 868/915MHz. A camada física do padrão IEEE802.15.4a passou a ser baseada em duas diferentes tecnologias [Karapistoli et al. 2010] [De Nardis and Di Benedetto 2007]:

- ***Utra Wideband (UWB) impulse radio*** - que opera em um espectro não licenciado UWB. Suporta três bandas de frequência independentes:
  - Banda Sub-GigaHertz - um único canal (canal 0) ocupando o espectro de 249,6 até 749,6MHz.
  - *Low Band* - consiste em quatro canais (canais de 1 a 4) ocupando o espectro de 3.1 a 4.8GHz.
  - *High Band* - consiste em 11 canais (canais de 5 a 15) ocupando o espectro de 5.8 a 10.6GHz.
- ***Chirp Spread Spectrum (CSS)*** - baseado no espectro não licenciado de 2.4GHz. Foi criado para suportar enlaces de longo alcance e para dispositivos móveis movendo-se em alta velocidade. A taxa de dados da camada física CSS passa a ser de 1Mbps (nominal) e de 250Kbps (opcional).

3. Ataque *Jamming*

O *Jamming* é definido como sendo a transmissão de um sinal de rádio com o intuito de interferir nas comunicações de uma rede de radiofrequência (RF) de forma intencional [Adamy 2004]. Existe uma outra forma de interferência conhecida como interferência de radiofrequência (em Inglês, RFI - *radio frequency interference*), que ocorre devido

IEEE 802.11		IEEE 802.15.4	
Canal	**Freq. (GHz)**	**Canal**	**Freq. (GHz)**
1	2.401 - 2.423	11	2.405
2	2.404 - 2.428	12	2.410
3	2.411 - 2.433	13	2.415
4	2.416 - 2.438	14	2.420
5	2.421 - 2.443	15	2.425
6	2.426 - 2.448	16	2.430
7	2.431 - 2.453	17	2.435
8	2.436 - 2.458	18	2.440
9	2.441 - 2.463	19	2.445
10	2.446 - 2.468	20	2.450
11	2.451 - 2.473	21	2.455
Os canais 22 a 26 do padrão IEEE 802.15.4 não sofrem interferência			

Tabela 2. Tabela de Canais Sobrepostos do Padrão IEEE 802.11 e IEEE 802.15.4

a proximidade a outras redes de RF que transmitem no mesmo canal de comunicação. Conforme a Tabela 2, redes no padrão IEEE 802.15.4 podem sofrer interferência de redes IEEE 802.11, pois alguns canais de frequências na banda de 2.4GHz são coincidentes.

3.1. Tipos de Ataques *Jamming* e Dispositivos *Jammers*

Com base nas características físicas descritas do padrão IEEE 802.15.4, podemos definir que um nó sensor pode se conectar a uma rede utilizando um número limitado de canais de comunicação: 16 canais na faixa de 2,4 GHz (2400-2483,5 MHz), 10 canais (30 para 2006) em 902-928 MHz e 1 canal (3 para 2006) em 868,3 MHz. Levando ainda em consideração que a potência máxima é de geralmente 0 dBm, pode-se perceber que um atacante pode facilmente interferir no sinal de uma rede de sensores.

O ataque de interferência tem seu sucesso ligado diretamente a relação sinal ruído de uma transmissão. O ruído nada mais é do que a representação da flutuação do espectro eletromagnético dada pela relação SNR = P_{signal}/P_{noise}, onde P representa a potência média. Um ataque *Jamming* se torna efetivo para SNR$<$1. Existem vários métodos de ataque de *Jamming* [O'Flynn 2011], sendo eles:

- ***Jamming de pontos*** - Neste tipo de ataque, toda a potência do sinal é dirigido para apenas um canal de transmissão. Assim, o sinal original é substituído pelo sinal do atacante. Para este tipo de ataque a melhor solução seria a mudança de canal de transmissão da rede.
- ***Jamming de varredura*** - Este ataque utiliza saltos de canais, ao invés de um canal só. O sinal atacante muda de canal alternadamente, ocupando assim toda a banda de transmissão.
- ***Jamming de barragem*** - O bloqueio é feito em intervalos de frequência, ou seja, o sinal de interferência é transmitido em dois ou mais canais ao mesmo tempo. Este ataque pode bloquear múltiplas frequências.
- ***Jamming enganador*** - Pode ser aplicado a uma única frequência ou conjunto de frequências. Este ataque é usado quando o atacante não quer ser detectado.

Ele é feito inundando a rede com dados falsos, com o intuito de causar falhas de transmissão e perda de pacotes.

Os dispositivos usados para gerar as interferências são chamados de *Jammers*. Normalmente, são equipamentos de RF simples, mas que transmitem o sinal em alta potência. Existem vários tipos de *jammers*. Xu et al. [Xu et al. 2005], propõem vários modelos genéricos, sendo eles:

- **Constante** - Emite um sinal aleatório e constante de alta potência. Este *jammer* envia apenas *bits* aleatórios. Sua função é manter o canal ocupado, interrompendo as transmissões no canal.
- **Enganador** - Transmite um sinal usando as técnicas do *jamming* enganador, isto é, pode transmitir dados em uma única frequência ou em um conjunto de frequências.
- **Aleatório** - Este tipo de jammer alterna aleatoriamente seu estado de repouso e de ataque a rede. Seu tipo de ataque a rede também pode ser qualquer um dos mencionados acima.
- **Reativo** - Escuta o canal de transmissão e ao detectar uma transmissão emite um sinal constante e aleatório para interferir na rede.

4. Contramedida ao Ataque de *Jamming* usando Rádio Cognitivo

O Rádio cognitivo pode analisar as frequências e o ambiente em que está conectado e com isto alterar as suas características de transmissão e recepção, tais como frequência e potência do sinal, melhorando a experiência do usuário [Mitola and Maguire 1999]. Um rádio cognitivo pode ser implementado em um dispositivo conhecido como *Software Defined Radio* (SDR). Define-se SDR como sendo um dispositivo que consegue alterar suas características de transmissão e recepção via software, reduzindo assim o número de funções realizadas em hardware [Jondral 2005].

Em Alberti et al. [Alberti et al. 2017] é proposta uma arquitetura de rádio cognitivo que analisa e armazena as informações de uso do espectro de frequência da rede publicando assim os canais disponíveis e ocupados. Nossa proposta se baseia nesta arquitetura. Nela, um rádio cognitivo com um sistema SDR analisa o espectro de radiofrequência, identifica qual o canal com menor interferência e redireciona automaticamente a transmissão e recepção dos nós sensores e dos roteadores de borda para este novo canal. Este sistema é composto por um módulo de célula de detecção e um dispositivo de rede sem fio. Conforme a Figura 2, o módulo de célula de detecção é composto por um SDR chamado de HackRF One™ para detectar o espectro de frequência e um laptop com middleware para processar sinais e realizar o controle SDR. O HackRF One opera de 1 MHz a 6 GHz e possui 20 MHz de largura de banda.

Exemplificando, de acordo com a Figura 3, o sistema de sensoriamento ligado ao notebook, tem a função de analisar as frequências de transmissão do padrão IEEE 802.15.4. As informações de cada canal são armazenadas e analisadas pelo algoritmo de sensoriamento, detectando assim o canal de menor interferência nos canais de transmissão. Assim, o sistema envia um comando aos sensores e ao roteador de borda. Este comando teria como função alterar a frequência de transmissão da rede de sensores para a frequência de menor interferência.

Figura 2. Hardware usado para o modulo de célula

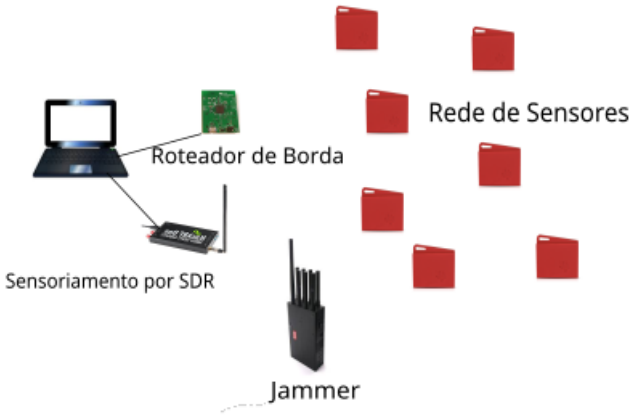


Figura 3. Exemplo de uma Rede de Sensores controlada pelo Sistema de Sensoriamento

4.1. Arquitetura do Sistema de Rádio Definido por Software

O SDR é controlado por um software que terá a função de detectar o melhor canal de transmissão. Este software opera através da plataforma GNU Radio, que é um conjunto de ferramentas usadas para a implementação de rádios definidos por software. Conforme o fluxograma definido na Figura 4, o algoritmo do GNU Radio transforma o fluxo de dados em vetores de 1024 posições, que são convertidos via bloco FFT para o domínio de frequência. Após esta conversão, calcula-se o módulo do fluxo gerado, que é convertido de tensão para energia em escala logarítmica.

Figura 4. Fluxograma do algoritmo de sensoriamento.

Para varrer os canais do padrão IEEE 802.15.4, um *script* executa o algoritmo de sensoriamento executando as trocas de frequência de 5 em 5 MHz, conforme a banda de canal do padrão. Os valores das amostras dos canais são armazenados em arquivos binários. Um outro *script* então, irá realizar os cálculos de energia do canal e trocar o canal dos nós sensores e dos roteadores de borda para o canal com menor energia detectado. A Figura 5 demonstra o fluxograma de funcionamento do *script* de sensoriamento.

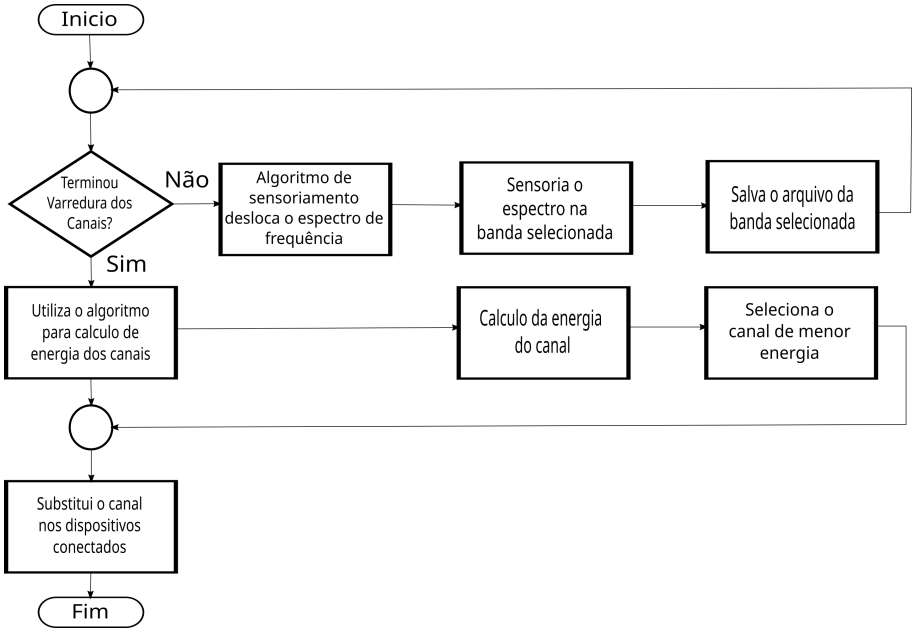


Figura 5. Fluxograma completo do sistema.

5. Trabalhos Relacionados

Um grande número de trabalhos de pesquisa sugerem soluções ao ataque de bloqueio *(Jamming)*, tendo como base os hardwares usados nas redes de sensores [Mpitziopoulos et al. 2009, Muraleedharan and Osadciw 2006, Xu et al. 2006]. Outros trabalhos sugerem novos designs para o nó como forma de combate ao ataque [Mpitziopoulos et al. 2007]. Em DeBruhl e Tague, é proposto como solução ao ataque a inclusão de um filtro digital nos receptores que teriam a função de filtrar o espectro de RF gerado pelos bloqueadores [DeBruhl and Tague 2011]. Em Mpitziopoulos et al. são discutidas várias técnicas de solução tendo como base o espalhamento espectral com saltos de frequência (em Inglês, FHSS - *Frequency-hopping spread spectrum*) e o espalhamento espectral de sequência direta (em Inglês, DSSS - *Direct Sequence Spread Spectrum*), sendo estas consideradas pelos autores as melhores soluções para as interferências de frequência [Mpitziopoulos et al. 2007]. Em Mpitziopoulos et al. é proposto o algoritmo JAID (*Jamming Avoidance Itinerary Design*), baseado na tecnologia conhecida como MA (*Mobile Agent*) proposta na literatura de RSSF. Este algoritmo calcula rotas quase ótimas para os MAs que fundem os dados de forma incremental ao visitar os nós e diante de ataques de bloqueio contra a WSN, modificam as rotas das MAs ignorando a área de interferência. Isso sem interromper a disseminação eficiente de dados pelos sensores [Mpitziopoulos et al. 2009]. Em Xu et al. são abordadas duas técnicas de proteção ao ataque de interferência. A primeira técnica conhecida como evasão de canal, visa sair do canal que o invasor está atacando e a outra visa competir com o atacante, ajustando alguns valores da rede, como potência de transmissão e codificação [Xu et al. 2006]. Em Muraleedharan e Osadciw, é proposto um novo mecanismo de segurança contra ataques de DOS. Este mecanismo é baseado em um algoritmo denominado *Ant System* que detecta o ataque de interferência nos nós sensores [Muraleedharan and Osadciw 2006]. Em Wood et al. é apresentado como solução a ataques de interferência o DEEJAM (*Defeating Energy-Efficient Jamming)*, um novo protocolo de camada MAC para o padrão IEEE

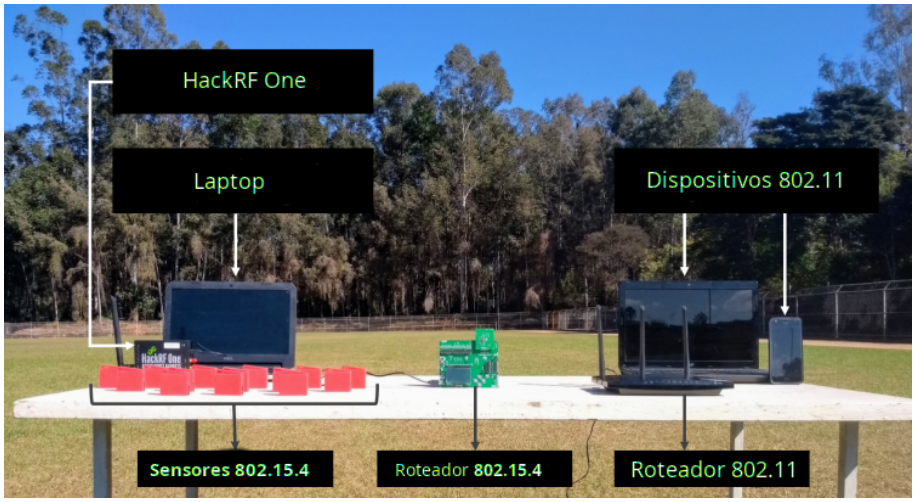


Figura 6. Cenário utilizado para o experimento.

802.15.4 [Wood et al. 2007]. No trabalho de Hamieh e Ben-Othman, os autores propõem como forma de detecção do ataque de interferência um novo método que se baseia na medição da distribuição de erros na rede [Hamieh and Ben-Othman 2009].

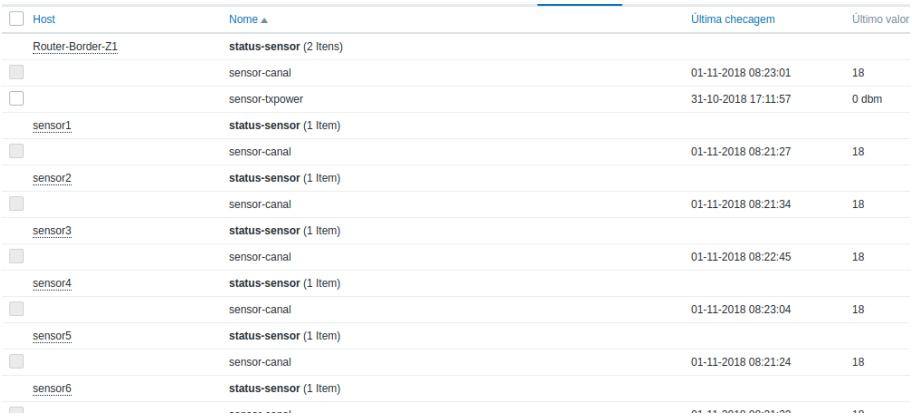
Nossa proposta de difere das demais, pois nosso sistema roda de maneira autônoma na rede, como um dispositivo a parte, que analisa todas as frequências sem assim gerar pacotes e filtros desnecessários na RSSF. Com isto, não afetando de maneira alguma a latência e não levando a perdas de pacotes.

6. Resultados Experimentais e Análise

A RSSF considerada neste experimento, ilustrada na Figura 6, opera no padrão IEEE 802.15.4. Ela é composta por um roteador de borda SMARTRF06EBK™ ligado a um laptop rodando o programa de gerência de rede Zabbix, que tem como função monitorar os canais da rede de sensores e caso ocorra alguma alteração disparar uma notificação de mudança de dados e sensores CC2650-sensortag™ fabricados pela Texas Instruments. Como Jammer é utilizado um roteador TP-Link N750 OpenWRT e dois laptops que se comunicam via canal 8, criando tráfego no canal de transmissão. O experimento foi executado em campo aberto, evitando assim outras interferências de redes sem fio.

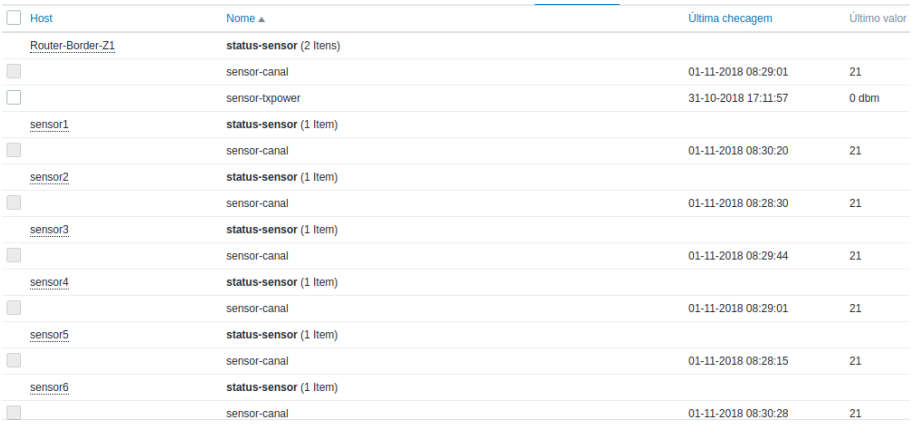
O roteador de borda e os sensores rodam o sistema operacional Contiki[Dunkels et al. 2004] e operam no canal 18. O canal 18 do padrão IEEE 802.15.4 e o canal 8 do padrão IEEE 802.11 são canais sobrepostos, gerando assim ruido de interferência de uma rede sobre a outra. O HackRF One está ligado ao laptop, onde está sendo executado o algoritmo de sensoriamento e análise de interferência. O laptop também está conectado ao roteador de borda da rede 802.15.4 para que possa ser coletado os dados para o Zabbix e para ser enviado o comando de troca de canal ao sensores e roteador de borda.

No inicio do experimento somente a rede de sensores está funcionando para que se possa levantar os dados do roteador e dos sensores no programa Zabbix, conforme mostra a Figura 7. Após alguns minutos, a rede 802.11 foi ligada, e foi gerado tráfego entre seus dispositivos para ser criado um aumento de interferência. Sem a interferência



Host	Nome ▲	Última checagem	Último valor
Router-Border-Z1	**status-sensor** (2 Itens)		
	sensor-canal	01-11-2018 08:23:01	18
	sensor-txpower	31-10-2018 17:11:57	0 dbm
sensor1	**status-sensor** (1 Item)		
	sensor-canal	01-11-2018 08:21:27	18
sensor2	**status-sensor** (1 Item)		
	sensor-canal	01-11-2018 08:21:34	18
sensor3	**status-sensor** (1 Item)		
	sensor-canal	01-11-2018 08:22:45	18
sensor4	**status-sensor** (1 Item)		
	sensor-canal	01-11-2018 08:23:04	18
sensor5	**status-sensor** (1 Item)		
	sensor-canal	01-11-2018 08:21:24	18
sensor6	**status-sensor** (1 Item)		

Figura 7. Tela inicial do experimento: sensores e roteador no canal 18.



Host	Nome ▲	Última checagem	Último valor
Router-Border-Z1	**status-sensor** (2 Itens)		
	sensor-canal	01-11-2018 08:29:01	21
	sensor-txpower	31-10-2018 17:11:57	0 dbm
sensor1	**status-sensor** (1 Item)		
	sensor-canal	01-11-2018 08:30:20	21
sensor2	**status-sensor** (1 Item)		
	sensor-canal	01-11-2018 08:28:30	21
sensor3	**status-sensor** (1 Item)		
	sensor-canal	01-11-2018 08:29:44	21
sensor4	**status-sensor** (1 Item)		
	sensor-canal	01-11-2018 08:29:01	21
sensor5	**status-sensor** (1 Item)		
	sensor-canal	01-11-2018 08:28:15	21
sensor6	**status-sensor** (1 Item)		
	sensor-canal	01-11-2018 08:30:28	21

Figura 8. Tela final do experimento: sensores e roteador no canal 21.

do Wi-Fi AP, a taxa de transferência do IEEE 802.15.4 foi de 1120 bps. Após a operação simultânea do Wi-Fi no canal 8, a taxa de transferência do 802.15.4 diminuiu para 520 bps. O sistema de sensoriamento então trocou automaticamente a rede 802.15.4 do canal 18 para o 21, conforme Figura 8. Para confirmação da troca de canais, a Figura 9 mostra os eventos de incidente gerados pelo Zabbix em relação aos canais do roteador e dos sensores.

7. Conclusão

Neste artigo, propomos e testamos um esquema de sensoriamento, utilizando Rádio Cognitivo sobre um sistema de SDR, para redes de sensores no padrão IEEE 802.15.4 , com o intuito de proteger a rede contra ataques de *Jamming*. Nos testes em rede real, que utilizaram sensores CC2650 da Texas Instruments, o sistema se mostrou satisfatório no auxilio ao combate dos seguintes tipos de *Jamming*: pontos, varredura e barragem. Nestes ataques o invasor direciona seu ataque para uma frequência específica ou para um grupo de frequências. Em nosso teste, nosso sistema pode detectar sempre qual a melhor frequência e executar a alteração dos sensores e do roteador de borda. Não houve em nenhum dos testes uma mudança no sistema de latência da rede e nem um aumento de perda de pacotes válidos, apenas, uma perda de acesso do roteador de borda aos sensores durante a troca dos canais.

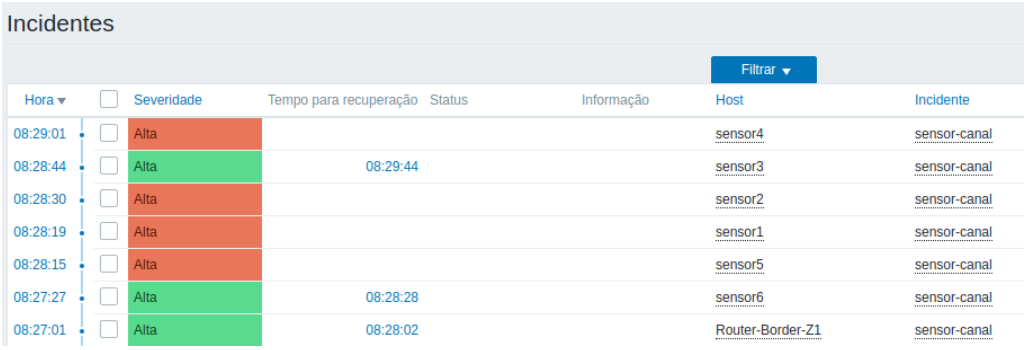


Figura 9. Tela de incidentes do Zabbix relatando mudança de canal dos sensores.

Agradecimentos

Este trabalho foi parcialmente financiado pela RNP, com recursos do MCTIC, processo No 01250.075413/2018-04, sob o projeto Centro de Referência em Radiocomunicações (CRR) do Instituto Nacional de Telecomunicações – Inatel, Brasil. Os autores agradecem também a FAPEMIG, CNPq e CAPES.

Este estudo foi parcialmente financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Referências

(2015). *IEEE Std 8 02.15.4TM TM-2 2015 - Revision of IEEE Std 802.15.4-2011*. IEEE.

Adamy, D. (2004). *EW 102: a second course in electronic warfare*. Artech House.

Alberti, A. M., Mazzer, D., Bontempo, M., de Oliveira, L. H., da Rosa Righi, R., and Sodré Jr, A. C. (2017). Cognitive radio in the context of internet of things using a novel future internet architecture called novagenesis. *Computers & Electrical Engineering*, 57:147–161.

De Nardis, L. and Di Benedetto, M.-G. (2007). Overview of the ieee 802.15. 4/4a standards for low data rate wireless personal data networks. In *Positioning, Navigation and Communication, 2007. WPNC'07. 4th Workshop on*, pages 285–289. IEEE.

DeBruhl, B. and Tague, P. (2011). Digital filter design for jamming mitigation in 802.15. 4 communication. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6. IEEE.

Dunkels, A., Gronvall, B., and Voigt, T. (2004). Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE.

Ergen, S. C. (2004). Zigbee/ieee 802.15. 4 summary. *UC Berkeley, September*, 10:17.

Hamieh, A. and Ben-Othman, J. (2009). Detection of jamming attacks in wireless ad hoc networks using error distribution. In *Communications, 2009. ICC'09. IEEE International Conference on*, pages 1–6. IEEE.

Jondral, F. K. (2005). Software-defined radio: basics and evolution to cognitive radio. *EURASIP journal on wireless communications and networking*, 2005(3):275–283.

Karapistoli, E., Pavlidou, F.-N., Gragopoulos, I., and Tsetsinas, I. (2010). An overview of the ieee 802.15. 4a standard. *IEEE Communications Magazine*, 48(1).

Misic, J. and Misic, V. (2008). *Wireless personal area networks: Performance, interconnection, and security with IEEE 802.15. 4*, volume 1. John Wiley & Sons.

Mitola, J. and Maguire, G. Q. (1999). Cognitive radio: making software radios more personal. *IEEE personal communications*, 6(4):13–18.

Mpitziopoulos, A., Gavalas, D., Konstantopoulos, C., and Pantziou, G. (2009). Jaid: An algorithm for data fusion and jamming avoidance on distributed sensor networks. *Pervasive and Mobile Computing*, 5(2):135–147.

Mpitziopoulos, A., Gavalas, D., Pantziou, G., and Konstantopoulos, C. (2007). Defending wireless sensor networks from jamming attacks. In *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, pages 1–5. IEEE.

Muraleedharan, R. and Osadciw, L. A. (2006). Jamming attack detection and countermeasures in wireless sensor network using ant system. In *Wireless Sensing and Processing*, volume 6248, page 62480G. International Society for Optics and Photonics.

O'Flynn, C. P. (2011). Message denial and alteration on ieee 802.15. 4 low-power radio networks. In *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, pages 1–5. IEEE.

Torrieri, D. J. (1985). *Principles of secure communication systems*. Artech House, Inc.

Wood, A. D. and Stankovic, J. A. (2002). Denial of service in sensor networks. *computer*, 35(10):54–62.

Wood, A. D., Stankovic, J. A., and Zhou, G. (2007). Deejam: Defeating energy-efficient jamming in ieee 802.15. 4-based wireless networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on*, pages 60–69. IEEE.

Xu, W., Ma, K., Trappe, W., and Zhang, Y. (2006). Jamming sensor networks: attack and defense strategies. *IEEE network*, 20(3):41–47.

Xu, W., Trappe, W., Zhang, Y., and Wood, T. (2005). The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 46–57. ACM.

HonIoT: Arquitetura de Honeynet com Controle de Propagação de Malwares para Dispositivos de IoT

Douglas B. de Godoy[1], Jaime F. Souza[1], Emerson R. A. Barea[2] e César A. C. Marcondes[3]

[1]Departamento de Computação (DC) - Universidade Federal de São Carlos (UFSCar)
[2]Departamento de Informática (DInf) - Instituto Federal do Tocantins (IFTO)
[3]Divisão de Ciência da Computação (IEC) - Instituto Tecnológico de Aeronáutica (ITA)

{douglas.godoy,jaime.souza}@ufscar.br, emerson.barea@ifto.edu.br, cmarcondes@ita.br

Resumo. *É indicada a utilização de honeypots e honeynets para o estudo aprofundado do comportamento dos malwares para dispositivos de IoT, porém, alguns cuidados devem ser tomados garantindo que esses ambientes sejam capazes de detalhar corretamente todo ciclo de infecção dos malwares e não se tornem um ponto de origem de ataques às outras redes. Nessa linha, este trabalho apresenta o HonIoT, uma honeynet que suporta a inclusão de honeypots reais, emuladas ou virtualizadas, criando uma internet emulada que corresponde à parte atacada da rede, possibilitando o monitoramento completo de todo ciclo de infecção e propagação do malware. Testes preliminares apontaram boa capacidade de escalabilidade do ambiente e correta execução do fluxo de ações da honeynet.*

Abstract. *It is indicated the use of honeypots and honeynets for IoT devices malware behavior study in-depth, but some attention should be taken to ensure these environments are able to accurately detail every malware infection cycle and do not become a attacker to other networks. In this line, this work presents HonIoT, a honeynet that supports inclusion of real, emulated or virtualized honeypots, creating an emulated internet corresponding to the attacked part of the network, allowing the complete monitoring of the whole cycle of malware infection and propagation. Preliminary tests pointed to good environment scalability and correct execution of honeynet's stock flow.*

1. Introdução

Nos últimos tempos foi perceptível o aumento da utilização de dispositivos de Internet das Coisas (*Internet of Things* - IoT) em áreas até então pouco exploradas. O desenvolvimento dessa estrutura tem sido bastante impulsionada principalmente pela evolução tecnológica de aparelhos de uso cotidiano, como telefones e TVs, porém, a IoT é mais ampla e atualmente está presente em diversas áreas, como indústrias, medicina, serviços, automobilismo, entre outras [Simpson et al. 2017] e [GARTNER 2017].

A ampliação da utilização de dispositivos de IoT veio acompanhada da necessidade de conectá-los às redes, privadas ou até mesmo pela internet, possibilitando sua utilização e gerenciamento remotos. Quanto a segurança, muitos desses dispositivos apresentam vulnerabilidades, que vão desde ausência de mecanismos de autenticação habilitados por padrão, passando por falhas em códigos, até problemas críticos de hardware [Dowling et al. 2017],

que vêm sendo exploradas por métodos diversos [SPAMHAUS 2018] e com efeitos cada vez mais abrangentes e críticos [Bhunia and Gurusamy 2017] [Marzano et al. 2018].

O cenário apresentado aponta claramente que o aumento da utilização de dispositivos de IoT e o risco associado a esse tipo de ambiente são problemas relevantes de pesquisa. Uma abordagem bastante utilizada para coleta e análise de malwares são os *honeypots* e *honeynets* ([Sochor and Zuzcak 2014], [Koniaris et al. 2014] e [Pathan 2014]), porém, esses mesmos ambientes podem não ser capazes de capturar corretamente todo ciclo de infecção de um *malware* ou até passar a ser um ponto de origem de ataque às outras redes caso não tomados alguns cuidados [Agnaou et al. 2016] e [Cabaj et al. 2017].

Baseado no exposto, esse trabalho propõe uma arquitetura de *honeynet* modular com controle de propagação de *malwares* para dispositivos de IoT. Entre os destaques e aspectos mais relevantes, essa *honeynet (i)* suporta a emulação de arquiteturas próprias de dispositivos de IoT com o uso de tecnologias de virtualização leve, lhe conferindo a possibilidade de ser livremente replicada em uma infraestrutura de computação genérica; *(ii)* emula a parte atacada da internet, portanto, não propagando ataques às outras redes enquanto fornece um ambiente propício para a completa infecção por parte do *malware*; e *(iii)* suporta o uso de equipamentos reais, emulados ou softwares de simulação como *honeypot*.

O restante desse trabalho está organizado da seguinte forma: a seção 2 apresenta os trabalhos relacionados ao tema principal da pesquisa. A seção 3 apresenta os requisitos a serem atendidos e os desafios impostos à proposta apresentada. Na seção 4 são apresentadas a arquitetura da *honeynet* e o detalhamento do protótipo desenvolvido. A seção 5 apresenta e comenta os resultados da avaliação do protótipo; e a seção 6 conclui e apresenta sugestões de trabalhos futuros.

2. Trabalhos Relacionados

São diversas as implementações de *honeypots* para dispositivos de IoT. Abaixo segue uma lista, não exautiva, de soluções divididas em algumas áreas de atuação.

O IoTPot [Pa et al. 2015] é um *honeypot* de baixa interatividade que emula *banners* reproduzidos artificialmente de diversos equipamentos de IoT utilizado para análise de tentativas de ataques por Telnet. Durante a fase de testes, os autores observaram mais de 76 mil tentativas de download de binários originados a partir de mais de 16 mil endereços IPs distintos. Adicionalmente, os autores fizeram o download de 43 exemplares de *malwares* para análise e constataram que eles estavam relacionados a 11 arquiteturas diferentes de CPU. Apesar dos resultados obtidos, os *banners* publicados pelo IoTPot são gerados manualmente, dificultando sua evolução e facilitando a identificação do *honeypot* pelo *malware*, já que não utiliza sistemas reais. Da mesma forma, ele não é capaz de detalhar o ciclo de infecção do *malware* apresentando as chamadas de rede e de sistemas nos alvos atacados.

Outra implementação existente é o HoneyThing [Erdem et al. 2018]. *Honeypot* de baixa interatividade que emula o protocolo TR-069, utilizado para gerenciamento de equipamentos de clientes (*Customer-Premises equipament* - CPE), como modens e roteadores domésticos. Também possui o servidor Web RomPager 4.07[1] simulando algumas

[1]https://www.allegrosoft.com/embedded-web-server-ae

vulnerabilidades associadas ao protocolo TR-069 encontrados em modens. Apesar de ser um *honeypot* específico para CPEs, por ser de baixa interatividade, o HoneyThing apenas gera estatísticas de tentativas de acessos sem detalhar o processo de infecção do dispositivo, como também não fornece mecanismos de contenção em caso de comprometimento do *honeypot*.

Mais uma implementação, agora com foco em redes de sensores sem fio (*Wireless Sensor Networks* - WSN), é o ZigBee *honeypot* [Dowling et al. 2017], que reproduz um dispositivo com suporte a rede ZigBee para identificar ataques ao SSH. Os autores identificaram nos testes realizados que o tipo de ataque com maior ocorrência estava relacionado a DDoS. Assim como os outros *honeypots* apresentados, o ZigBee *honeypot* é de baixa interatividade, possuindo as mesmas limitações das soluções anteriores.

Por fim, outro exemplo de implementação é o HIoTPOT [Gandhi et al. 2018], que consiste em um *honeypot* de alta interatividade baseado no Raspberry PI conectado a duas redes, uma de dispositivos reais e a outra de dispositivos falsos. O HIoTPOT possui uma base de dados onde estão registrados todos usuários válidos e autenticados na rede. Quando um usuário não autenticado tenta acessar outro equipamento da rede, o *honeypot* automaticamente direciona seu tráfego para os dispositivos da rede falsa, mantendo segura a rede real enquanto possibilita a análise do processo de infecção do *malware*. Apesar de eficiente, o HIoTPOT tem custo elevado com a duplicação da estrutura da rede física, enquanto seu método de identificação de ataque é simples e passível de falha.

Como citado anteriormente, apesar de não exaustiva, a lista de *honeypots* apresentados indica que as soluções normalmente são baseados em serviços e protocolos específicos, reduzindo muito seu escopo e funcionalidades. Da mesma forma, não possuem mecanismos de contenção contra propagação de ataques caso o ambiente seja comprometido. Por outro lado, o exemplo de *honeypot* de alta interatividade apresentado é baseado em ambiente físico, elevando seu custo, além de possuir um mecanismo de contenção contra propagação de *malwares* muito simples e ineficiente.

3. Requisitos e Desafios

Ao propor o desenvolvimento de um *honeypot* de alta interatividade, ou *honeynet*, com controle de propagação de *malwares* para dispositivos de IoT emulados, alguns novos questionamentos surgem e levantam consigo requisitos ainda não contemplados.

Um *honeypot* de alta interatividade, assim como apresentado na Figura 1, tem como requisitos principais que as aplicações e serviços oferecidos por ele sejam reais, que possam ser comprometidos pelo *malware* e que o atacante tenha acesso irrestrito ao sistema operacional (S.O.) do dispositivo. Esses requisitos garantem que o *malware* não identifique que está em um *honeypot*, infectando completamente o ambiente e viabilizando a análise de seu funcionamento. Quanto aos desafios, são caracterizados pelo alto custo, dificuldade e complexidade de instalação e manutenção do ambiente, além do alto risco de comprometimento da estrutura e consequente propagação do *malware* utilizando o próprio *honeypot* como um ponto de ataque às outras redes [Hoepers et al. 2007].

Já os *honeypots* de baixa interatividade são reconhecidos por seu baixo custo de instalação, manutenção e risco de comprometimento, porém, como apresentado na seção 2, os resultados obtidos com sua utilização são limitados principalmente pelas deficiências

Características	*Honeypot* de baixa interatividade	*Honeypot* de alta interativ./*Honeynet*
Instalação	**fácil**	mais difícil
Manutenção	**fácil**	trabalhosa
Risco de comprometimento	**baixo**	alto
Obtenção de informações	limitada	**extensiva**
Necessidade de mecanismos de contenção	**não**	sim
Atacante tem acesso ao S.O. real	não (em teoria)	**sim**
Aplicações e serviços oferecidos	**emulados e reais**	
Atacante pode comprometer o *honeypot*	**não (em teoria)**	sim

Figura 1. Características desejáveis ao *honeynet* com controle de propgação de *malwares* para dispositivos de IoT. Fonte: adaptado [Hoepers et al. 2007].

de implementação ou ausência de recursos existentes nos sistemas reais, possibilitando serem descobertos pelos *malwares* mais avançados.

Baseado nas características apresentadas, identificamos que a proposta mais adequada para este trabalho é unir as características mais vantajosas dos dois tipos de *honeypots* (de baixa e alta interatividade), conforme destacado na Figura 1. Fazendo uso de sistemas reais de dispositivos de IoT em arquitetura correspondente emulada com o auxílio de softwares específicos, reduzindo o custo, simplificando a instalação e manutenção do ambiente; e emulando a parte atacada da rede com todas suas máquinas e serviços, impedindo a propagação do *malware* às outras redes enquanto cria um ambiente completo para infecção do *malware*, proporcionando meios para captura de informações para seu estudo.

4. Materiais e Métodos

Nesta seção são detalhados os elementos formadores da arquitetura da *honeynet* (seção 4.1), seguido da apresentação da estrutura física utilizada no desenvolvimento do protótipo HonIoT (seção 4.2) e do detalhamento do fluxo de ações e códigos do controlador do protótipo (seção 4.3).

4.1. Arquitetura

Para atingir os objetivos propostos, atendendo aos requisitos apresentados e solucionando os desafios impostos, foi definida uma arquitetura modularizada para *honeynet* de dispositivos de IoT. Essa arquitetura é responsável por suportar a análise de todo ciclo de infecção e propagação de *malwares* para IoT, com estrutura de *honeypot* de alta interatividade com controle de propagação, expondo totalmente um sistema de IoT aos ataques provindos da internet, enquanto monitora todo acesso e fluxo de dados num ambiente controlado, criando a representação de parte da internet quando o *honeypot* passa a ser um ponto de origem e disseminação de ataques.

Para facilitar a compreensão de sua estrutura e o funcionamento de cada componente, a arquitetura foi dividida em quatro módulos principais, conforme apresentado na Figura 2. Abaixo estão descritas as funcionalidades dos elementos pertencentes a cada módulo e a relação entre eles:

Dispositivos de IoT: corresponde aos dispositivos destinados a sofrerem ataques na *honeynet*. Suporta a utilização de *honeypots* de baixa interatividade, com reprodução de serviços ou sistemas de IoT virtualizados em *hardware* genérico ou com a utilização de plataformas e sistemas construídos especialmente para capturar informações de acessos maliciosos; ou alta interatividade, suportando a utilização de sistemas e dispositivos de IoT emulados em plataformas específicas ou reais conectados à *honeynet*.

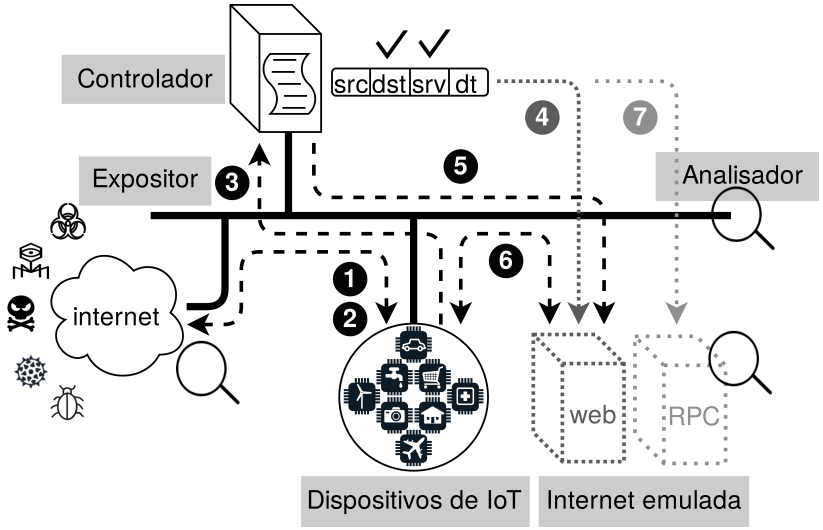


Figura 2. HonIoT: arquitetura da *honeynet* com controle de propagação de *malwares* para dispositivos IoT.

Outra função desse módulo é garantir que o *malware* não identifique que está em um *honeypot*, possibilitando a infecção completa do ambiente. Alguns *malwares* utilizam-se de artifícios variados para identificar se o ambiente é um *honeypot*, limintando ou até mesmo anulando seu comportamento quando a identificação é positiva. Dessa forma, é aconselhável utilizar plataformas e sistemas que reproduzam fielmente a original, com o auxílio de emulação da arquitetura correta e configurações adicionais necessárias, como a implementação de *banners* que correspondam aos dos serviços reais e a ocultação de parâmetros e informações do sistema base de emulação do ambiente.

Expositor: esse módulo corresponde a toda infraestrutura de rede necessária para exposição completa dos dispositivos de IoT (*honeypot*). Sua função é possibilitar que *malwares* provindos da internet acessem o *honeypot* para infectá-lo completamente. Essa fase corresponde aos Fluxos 1 e 2 apresentados na Figura 2 (fluxos identificados pelas bolas pretas com os números 1 e 2). No Fluxo 1, o *honeypot* é acessado livremente por qualquer *malware* com origem na internet, sem restrição alguma, possibilitando sua exposição completa e, consequentemente, aumentando a probabilidade de captura de informações de *malwares* diversos. Já o Fluxo 2 permite que o *honeypot* acesse os servidores de comando e controle (C&C) dos *malwares* distribuídos na internet, e vice-versa, a fim de completar o ciclo de infecção do *malware*.

Toda a exposição representada pelos Fluxos 1 e 2 na arquitetura são gerenciadas pelo $Controlador$ da *honeynet*, módulo detalhado em seguida.

Controlador: esse módulo é considerado o mais importante da *honeynet*, pois, é ele que garante a inteligência da rede atuando como um gerenciador de tráfego pró-

ativo, durante a infecção do *honeypot*; e também reativo ao comportamento do *malware*. Inicialmente sua função é permitir o fluxo de dados entre os *malwares* provindos da internet e o *honeypot*, como apresentado anteriormente no módulo $Expositor$, onde toda comunicação é baseada em regras de controle de tráfego e definição de rotas que permitem o encaminhamento dos dados entre os dispositivos envolvidos na comunicação.

A partir da infecção completa do *honeypot*, todo tráfego originado por ele deve ser interceptado pelo $Expositor$ e direcionado ao $Controlador$ (Fluxo 3), pois, a partir desse momento, os novos fluxos podem estar relacionados aos procedimentos de propagação e disseminação do *malware*, que devem ser evitados para não tornar a *honeynet* num ponto de origem de ataques. Ao mesmo tempo, a análise dessas chamadas de rede são material fundamental para o estudo e entendimento do comportamento e funcionamento do *malware*.

Com o tráfego interceptado, o $Controlador$ identifica e extrai os parâmetros de endereçamento de rede e do serviço solicitados das máquinas envolvidas na comunicação - endereço IP de destino, protocolo e portas de serviço utilizadas. Com posse dessas informações, o $Controlador$ tem a função de criar as máquinas da $Internet\ emulada$ sob demanda, criando automaticamente máquinas clone configuradas com os parâmetros de rede e de serviços previamente extraídos pelo $Controlador$ (Fluxo 4).

A partir da criação da máquina clone, o controlador cria as regras de roteamento e direcionamento de tráfego, instalando-as no $Expositor$ para então encaminhar os dados da solicitação de comunicação diretamente à máquina clone na $Internet\ emulada$ (Fluxo 5). A partir desse momento, toda nova comunicação entre o *honeypot* e a máquina clone é feita diretamente através do ambiente $Expositor$ (Fluxo 6), dessa forma, nenhum tráfego originado no *honeypot* correspondente a uma tentativa de propagação do *malware*, ou qualquer outra ação dele, será propagada para outras redes. Esse procedimento impede que a *honeynet* seja um ponto de ataque e viabiliza o monitoramento do tráfego de rede originado pelo *malware*, possibilitando sua análise detalhada. A cada solicitação de comunicação iniciada pelo *honeypot* a um novo destino, protocolo ou serviço, esse processo é repetido (Fluxo 7).

Analisador: esse módulo é responsável por capturar dados que servirão como base de informações para a análise de todo processo de infecção, comprometimento e propagação do *malware*. As ferramentas pertencentes a esse módulo estão distribuídas em quatro áreas específicas da arquitetura:

- Internet: aqui o analisador monitora todas as tentativas de acesso ao *honeypot*, incluindo tempo, volumetria de tráfego, origem dos ataques e quais mecanismos são utilizados.
- Dispositivos de IoT: o monitoramento interno nos dispositivos de IoT provê informações que podem ser utilizadas para análise das ações do *malware* nesses dispositivos, através de monitoramento da utilização de memória, processos criados, chamadas de sistema ou outros parâmetros que se fizerem necessários. Vale ressaltar que o monitoramento nesse ponto dependente do tipo de *honeypot* utilizado, pois, devem ser respeitadas restrições de acesso a determinados parâmetros por questão de licenças ou limitações técnicas em equipamentos ou softwares proprietários.
- Rede da Internet emulada: monitora todo tráfego de rede passante entre o *honeypot*

e todos elementos da $Internet\ emulada$, assim como realizado no link de Internet. Monitora o fluxo de dados correspondente à propagação do *malware* a partir do comprometimento do *honeypot*.

- Máquinas da Internet emulada: aqui o analisador monitora as ações executadas pelo *malware* internamente nas máquinas da $Internet\ emulada$, como apresentado no Dispositivo de IoT. O objetivo é gerar informações que sirvam como base para análise do comportamento do malware nas máquinas atacadas durante sua propagação.

Com a utilização de todas informações obtidas no módulo $Analisador$, é possível realizar um estudo aprofundado de todo ciclo de infecção de um malware.

4.2. HonIoT

Para validar a arquitetura da *honeynet* com controle de propagação de *malwares* para dispositivos de IoT proposta neste trabalho e apresentada na subseção 4.1, foi desenvolvido o protótipo HonIoT (*Honeynet* IoT), cuja finalidade é simular parte do processo de infecção sofrido por um *honeypot* de alta interatividade a partir de *malwares* originados na internet, possibilitando o estudo dos mecanismos de infecção e propagação.

O HonIoT é um protótipo que une um conjunto de aplicações de rede e emulação leve pré-existentes, a elementos desenvolvidos especialmente para esse fim. De modo geral, baseia-se na plataforma de emulação Qemu versão 2.11.1 como base para o $Dispositivo\ de\ IoT$ (*honeypot*); o OVS versão 2.9.2 e Wireshark versão 2.6.5 na estrutura do $Expositor$; o LXD versão 3.0.3 como base para as máquinas da $Internet\ emulada$; e o Ryu versão 4.24 como controlador para SDN do OVS.

A estrutura física do HonIoT é composta por dois servidores físicos, cada um contendo um processador de 3.50GHz e 8 núcleos, 16GB de memória RAM e múltiplas interfaces de rede de 1Gbps cada. Além de um computador com processador de 3GHz e 4 núcleos, 2 GB de memória RAM e duas interfaces de rede de 1Gbps cada. As máquinas foram conectadas fisicamente conforme apresentado na Figura 3. Em todas as máquinas foi instalado o S.O. Ubuntu 16.04.4 LTS Server sem serviços adicionais, propiciando um ambiente livre de interferências indesejáveis.

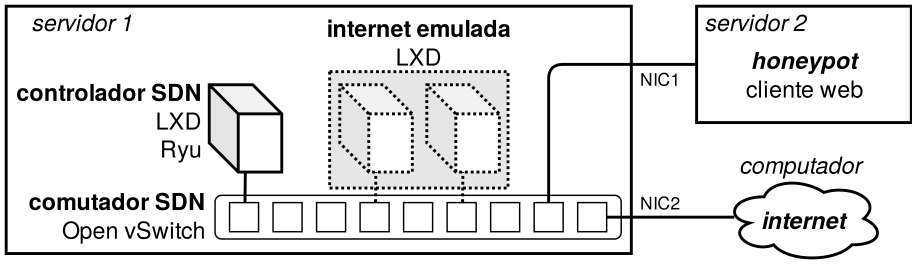


Figura 3. Topologia para o protótipo *HonIoT*.

Conforme apresentado na Figura 3, o $servidor\ 1$ hospeda todo ambiente de controle e emulação responsáveis por reproduzir a porção da rede atacada. Tal ambiente é composto por $controlador\ SDN$ (*container* LXD com Ryu instalado), $comutador\ SDN$ (OVS) e $internet\ emulada$ (*containers* LXD criados sob demanda). O $servidor\ 2$ possui o S.O. Linux instalado contendo softwares de servidor e cliente Web instalados sobre

uma arquitetura ARM emulada (Qemu) simulando um *honeypot*. Já o computador que representa a $internet$ possui apenas o S.O. Linux instalado diretamente em seu hardware.

O $servidor\ 1$ possui uma interface de rede conecta diretamente ao $servidor\ 2$. Essa interface também está conectada logicamente ao $comutador\ SDN$, possibilitando a comunicação entre o ambiente virtualizado da *honeynet* e o *honeypot* emulado. O $servidor\ 1$ também possui uma interface de rede conectada fisicamente ao computador $internet$ e logicamente ao $comutador\ SDN$, possibilitando a interligação de todos elementos virtuais e físicos no protótipo.

4.3. Fluxo de Ações

Quanto ao fluxo das ações executadas para validação do protótipo, inicialmente foram geradas requisições de rede com origem na $internet$ e destinadas ao servidor Web do $honeypot$, simulando a infecção por um *malware*. Essas chamadas são encaminhadas sem restrição no $comutador\ SDN$, ao mesmo tempo em que o tráfego de rede é monitorado ao passar pelo comutador. Essa etapa da validação do protótipo corresponde ao Fluxo 1 da Figura 2.

Para isso, assim que percebe o fluxo de dados, o $comutadorSDN$ intercepta e direciona os pacotes ao $controladorSDN$, que inspeciona o parâmetro correspondente ao tipo de protocolo da camada 4, identificando se o segmento é TCP ($bits == 2$ indica que o segmento possui a flag SYN utilizada). A partir dai, o $controladorSDN$ extrai os endereços IP de origem e destino do datagrama identificando se a origem está na $internet$ (src_port) e o destino é o *honeypot* ($HONEYPOT_IP$). Caso a comunicação atenda essa premissa, o $controladorSDN$ cria a regra que permite a comunicação entre a máquina da internet e o *honeypot* ($match1$) e a instala no $comutador\ SDN$ (add_flow) (Figura 4).

```
###
# comunicação internet <-> honeypot
if dst_ip == self.HONEYPOT_IP and \
((protocol == 'TCP' and bits == 2) or protocol != 'TCP'):
    ...
    match1 = msg.datapath.ofproto_parser.OFPMatch(eth_type=2048, \
        ip_proto=protocolN, udp_src=src_port, udp_dst=dst_port, \
        ipv4_src=src_ip, ipv4_dst=dst_ip)
    ...
    action1 = [msg.datapath.ofproto_parser.OFPActionOutput(out_port1)]
    ...
    self.add_flow(msg.datapath, 1, match1, action1)
```

Figura 4. Código Python do controlador que permite máquinas da internet acessarem livremente o *honeypot*.

Seguindo esse processo, quando o *honeypot* inicia a comunicação com um servidor de C&C, o tráfego também deve ser permitido, conforme descrito na seção 4.1, pois essa comunicação é considerada requisito para que o *malware* continue o processo de infecção. Durante esse processo, o $controladorSDN$ verifica se o endereço IP de destino contido no datagrama interceptado (dst_ip) existe na *white_list* (ip_list) recuperada em [Emerging Threats 2017]. Essa filtragem por endereços da *white_list* permite que o

controlador identifique quais endereços IP são de servidores de C&C, permitindo que o $controladorSDN$ crie regras liberando o tráfego para esses servidores (Figura 5).

```
###
# comunicação honeypot <-> C&C
def is_attack(self, dst_ip):
    if self.command_ip_list.in_list(dst_ip):
        return False
    else:
        return True
```

Figura 5. Código Python do controlador que permite a comunicação do *honeypot* com servidores de C&C.

Todo tráfego gerado no *honeypot* com destino a qualquer endereço IP não constante na *white list* pode representar uma tentativa de propagação do *malware* que deve ser impedida pelo HonIoT. Para isso, o $comutador\ SDN$ intercepta o fluxo de dados e o encaminha ao $controlador\ SDN$ que, por sua vez, extrai as informações de endereçamento das máquinas, protocolos e serviços contidos no fluxo, e acessa o gerenciador de *containers* localizado no *servidor 1* através de uma conexão SSH, solicitando a criação de um novo *container* LXD a partir de modelos pré-estabelecidos com base no serviço desejado ($DEFAULT_CONTAINER_TEMPLATE$) (Figura 6). A partir desse momento, o gerenciador de *containers* configura o *container* recém criado com os parâmetros de rede identificados anteriormente.

```
###
# honeypot atacando outras máquinas
    def create_container(self, msg, src_ethernet, ip_version, src_ip, \
                         dst_ip, container_name):
        template = self.DEFAULT_CONTAINER_TEMPLATE
        ...
        ssh = SSH()
        cmd = 'lxc launch %s %s' % (template, container_name)
        ssh.exec_cmd(cmd)
        ...
        cmd = 'lxc exec %s -- cat /sys/class/net/eth0/address' \
            % (container_name)
        new_container_mac = ssh.exec_cmd(cmd).strip()
```

Figura 6. Exemplo de código do controlador que cria *containers* na Internet emulada.

Feito isso, o $controladorSDN$ recupera a informação de endereço de camada 2 (*Media Access Control* - MAC) atribuído automaticamente ao *container* recém gerado pelo serviço LXD ($new_container_mac$), cria e instala uma regra ($action1$ e $action2$ - Figura 7) no $comutador\ SDN$ permitindo o tráfego entre o *honeypot* e o *container* com base nesse endereço, possibilitando assim o encaminhamento de tráfego diretamente entre essas máquinas.

```
action1 = [msg.datapath.ofproto_parser.\
          OFPActionSetField(eth_dst=new_container_mac), \
          msg.datapath.ofproto_parser.OFPActionOutput(out_port1)]
action2 = [msg.datapath.ofproto_parser.OFPActionOutput(out_port2)]
```

Figura 7. Código Python do controlador que cria e instala a regra para comunicação com o *container* da internet emulada.

Todo esse processo é repetido sempre que o *honeypot* gera fluxos de dados para novos destinos, protocolos ou serviços, impedindo que o *honeypot* propague o *malware*, além de viabilizar a análise de todo esse processo com o monitoramento dos fluxos de dados na rede e as ações executadas pelo *malware* nas máquinas da $internet\ emulada$.

5. Testes de Validação e Discussão dos Resultados

Para avaliar o completo funcionamento do protótipo HonIoT, dividimos os testes em 4 etapas principais que estão detalhadas a seguir:

Experimentos de Invasão ao honeypot: esse teste analisou o correto funcionamento do processo em que máquinas distintas da $internet$ acessam serviços disponíveis no *honeypot*, simulando uma tentativa de iniciar a infecção. Para isso, foram geradas 100 requisições ao serviço Web partindo de IPs gerados aleatoriamente na máquina da $internet$ destinadas ao *honeypot*. Nesse teste, 100% das requisições foram devidamente tratadas pelo $comutador\ SDN$ e $controlador\ SDN$, seguindo o procedimento apresentado na seção 4.3, Figura 4, com todas solicitações de acesso devidamente encaminhadas ao *honeypot*, conforme demonstraram os rastros de execução. Essa validação foi considerada satisfatória.

Experimentos de Comunicação *Honeypot* e Servidores de C&C: em seguida, um segundo experimento foi o de acompanhar o correto funcionamento da comunicação livre do *honeypot* com servidores de C&C a qualquer momento. Esse acesso deve ser permitido quando o endereço IP de destino do datagrama emitido pelo *honeypot* constar na *white list* de servidores de C&C. Ao invés de testar com um *malware* imprevisível, foram feitos testes de conexão iniciadas a partir do *servidor 2* (*honeypot*), para isso, foram gerados requisições Web originadas no *honeypot* e destinadas aos endereços IP dos servidores constantes na *white list* de servidores de C&C. Ao final do teste, observou-se que todas requisições foram devidamente tratadas pelo $comutador\ SDN$ e $controlador\ SDN$, e encaminhadas para a saída de $internet$ corretamente. Os rastros de execução demonstraram que esse teste de validação foi considerado satisfatório.

Experimentos de Criação das máquinas na Internet emulada: esse terceiro experimento teve o propósito de verificar se o $controlador\ SDN$ solicita corretamente a criação de um ou mais *containers* ao *Servidor 1* após receber datagramas IP com parâmetros de rede "não constantes" em regras já instaladas no $comutador\ SDN$; e se os *containers* são criados e inicializados corretamente para encaminhamento do tráfego partindo do *honeypot* em tempo hábil.

Nessa etapa, o interesse era em dois cenários possíveis. O primeiro verificou o comportamento do ambiente, na forma dessa implementação, quando solicitadas chamadas de grupos de *containers* de forma sequencial, onde um *container* posterior somente é solicitado quando o anterior já está totalmente funcional. A ideia era estudar o comportamento

de um *malware* que interage e analisa o ambiente em que está antes de tomar a próxima decisão.

O segundo cenário analisou o comportamento do ambiente quando grupos de *containers* são solicitados em rajadas pelo *honeypot*, simulando uma situação de ocorrência de *scan* de rede. O *scan* de rede pode ser intenso e requerer a criação de dezenas ou centenas de *containers* em cada execução. Esse cenário permitiu verificar e analisar a escalabilidade e estabilidade do protótipo nesse tipo de situação.

Os resultados demonstrados na Figura 8 apresentam a utilização de memória (em MB) agrupada por conjuntos de *containers* criados sequencialmente, ou seja, no cenário 1. Os dados mostram que a utilização de memória está diretamente relacionada ao número de *containers* criados, porém, seu crescimento não é linearmente perfeito, partindo de aproximados 70 MB para criação de 1 *container*, passando por aproximados 463 MB de memória consumida para um grupo de 50 *containers*, chegando a até 925 MB de memória consumida para grupos de 100 *containers*, sempre com 100% de sucesso na criação dos *containers*. Os resultados indicam uma sobrecarga de memória ao serem criados mais containers. Não foi investigado a fundo o porque desse comportamento, mas pode ser em função de replicação de dados do sistema operacional.

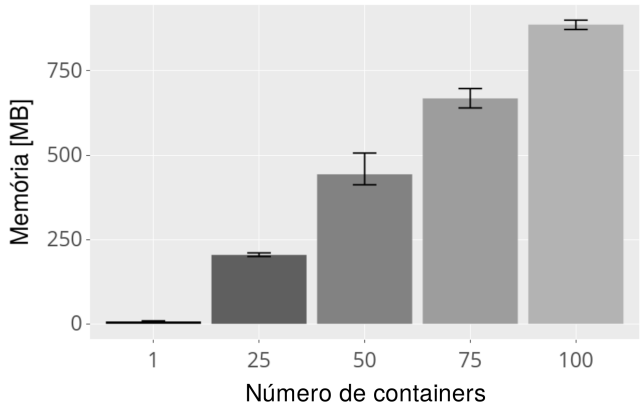


Figura 8. Memória (em MB) consumida para criação dos *containers* correspondentes às máquinas da *Internet emulada*.

Apesar desse comportamento de sobrecarga adicional por *container* adicional, o protótipo comprova que a utilização de *containers* LXD na $Internet\ emulada$ é promissora quando considerado o consumo total de memória versus o número de máquinas emuladas. Ou seja, 100 *containers* em uma máquina servidora pode ser considerado baixo uso de recursos.

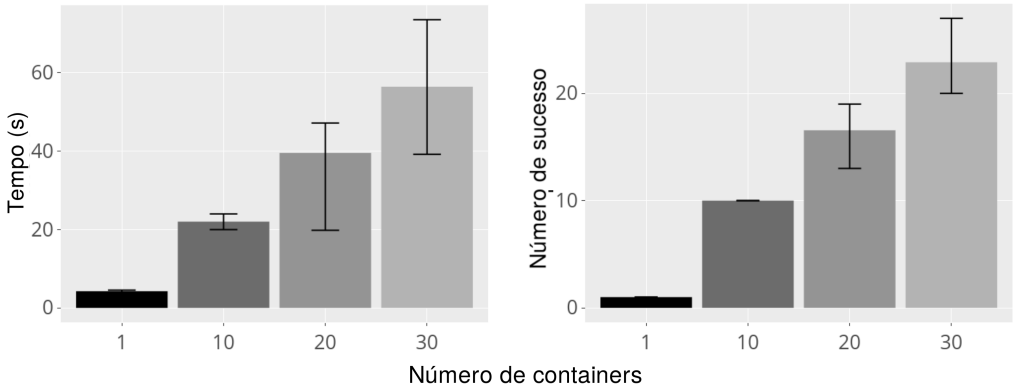
Outro resultado importante identificado é o comportamento padrão quanto ao tempo de criação dos *containers*, que foi de 4.1 segundos em média por *container*. Esse tempo tem o revés de ser elevado para tratar o encaminhamento de um pacote, podendo gerar desconfiança em um *malware* mais sofisticado sobre a possibilidade de ele estar em um *honeypot*.

Com relação ao cenário 2, onde é analisado o comportamento do protótipo frente a situações de rajadas de pacotes, foram verificados o tempo de criação para os grupos

de *containers* e a quantidade de sucesso nesse procedimento. Como não existe tempo mínimo entre as chegadas de pacotes, o $controlador\ SDN$ recebe múltiplas requisições simultâneas por meio de eventos em *threads*, e repassa para o gerenciador de *containers* as solicitações de criação no melhor esforço da tecnologia utilizada no protótipo.

Os dados da Figura 9a apresentam que, apesar das solicitações terem sido geradas em rajadas a partir do *honeypot*, onde verificou-se um tempo médio de apenas 0,6 segundos para o encaminhamento de 100% das solicitações, houve um aumento considerável de tempo para criação dos *containers* na $Internet\ emulada$ do *servidor 1*, chegando a aproximadamente 53 segundos em média para grupos de 30 *containers* solicitados. Esse comportamento ocorre devido os recursos de criação de *containers* serem compartilhados e a contenção ser grande.

Além dos tempos de instanciação serem elevados, outro dado importante é a grande variação do tempo obtido entre os testes, que demonstrou instabilidade do ambiente do protótipo nesse caso.



(a) Tempo de criação dos *containers*

(b) Número de *containers* criados com sucesso

Figura 9. Tempo (segundos) consumidos para criação de grupos de *containers* solicitados em rajadas (a) e quantidade de *containers* criados com sucesso por grupos de *containers* solicitados em rajadas (b).

Para analisar essa questão de instabilidade e a incapacidade de dar vazão rápida para um número grande de instanciação de *containers*, uma observação importante é o aumento expressivo da taxa de insucesso com o aumento da quantidade de *containers* solicitados nas rajadas. Conforme apresentado na Figura 9b, a taxa de insucesso é de aproximadamente 10% a partir da solicitação de criação de 20 *containers*, aumentando consideravelmente para até 20% de falhas para grupos de 30 *containers* solicitados.

Análises preliminares dos resultados apontam que as tecnologias utilizadas no protótipo possuem deficiência na forma como o controlador conecta no *servidor 1* para solicitar a criação dos *containers*, utilizando conexões SSH através do Python Paramiko. Os rastros da execução apontaram erros no estabelecimento de conexões entre essas duas máquinas quando as solicitações chegam em rajadas. Uma possível alternativa é a utilização de sockets Unix ou outro tipo de comunicação de rede menos complexa que uma conexão SSH. Uma outra opção é a pré-instanciação de centenas de *containers* no inicio da criação do ambiente experimental, cabendo ao controlador somente a tarefa de

configurá-los com os parâmetros de rede extraídos previamente durante a interceptação dos fluxos de dados.

Comunicação com as máquinas da Internet emulada: o último experimento de validação verificou o funcionamento do encaminhamento do tráfego partindo do *honeypot* e destinado aos *containers* da $Internet\ emulada$, ao invés de direcioná-lo para saída pelo link de internet real da *honeynet*. Nessa etapa, basicamente após a criação dos *containers* na $Internet\ emulada$, foi observado a criação as regras de encaminhamento no $comutador\ SDN$, atualizando a tabela de fluxos e, a partir dai, todo tráfego partindo do *honeynet* foi encaminhando corretamente, sem atrasos relevantes ou quebra de conexão TCP para as partes correspondentes. Os rastros de execução do controlador e dos *containers* envolvidos demonstraram que esse teste de validação foi considerado satisfatório.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou o HonIoT, uma *honeynet* para dispositivos de IoT que suporta *honeypots* baseados em equipamentos físicos e reais, simulados ou emulados. Garante a exposição completa do *honeypot* aos *malwares* disponíveis na internet, enquanto garante o controle de propagação do *malware* criando uma internet emulada e protegida, além de propiciar um ambiente completo de monitoramento das ações do *malware* através do monitoramento de suas ações no *honeypot*, na rede e nas máquinas emuladas.

Testes preliminares demonstraram bom nível de escalabilidade da *honeynet* quanto ao tempo de resposta às solicitações de criação de máquinas na rede emulada e quantidade suportada, bem como o correto funcionamento do fluxo de ações do protótipo proposto.

Como trabalhos futuros, considera-se relevante investigar maneiras mais sofisticadas para identificar quando o *malware* que infectou o *honeypot* pretende acessar servidores de C&C ou realizar um ataque, testar outras técnicas de conexão entre o $controlador\ SDN$ e a máquina hospedeira da $Internet\ emulada$ para melhorar a taxa de sucesso da criação de *containers* em situação de rajadas de fluxos de rede. Outra necessidade é testar o HonIoT em ambiente real para monitorar seu comportamento em produção.

Agradecimentos

Este trabalho teve apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 e do Instituto Federal do Tocantins (IFTO).

Referências

Agnaou, A., Kalam, A. A. E., Ouahman, A. A., and Montfort, M. D. (2016). Reduce positive and negative falses from attacks collected from the deployment of distributed honeypot network. *CoRR*, abs/1611.03252.

Bhunia, S. S. and Gurusamy, M. (2017). Dynamic attack detection and mitigation in iot using sdn. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6.

Cabaj, K., Gawkowski, P., Grochowski, K., Nowikowski, A., and Żórawski, P. (2017). The impact of malware evolution on the analysis methods and infrastructure. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 549–553.

Dowling, S., Schukat, M., and Melvin, H. (2017). Data-centric framework for adaptive smart city honeynets. In *2017 Smart City Symposium Prague (SCSP)*, pages 1–7.

Dowling, S., Schukat, M., and Melvin, H. (2017). A zigbee honeypot to assess iot cyberattack behaviour. In *2017 28th Irish Signals and Systems Conference (ISSC)*, pages 1–6.

Emerging Threats (2017). Emerging threats botnet command and control drop rules. `http://rules.emergingthreats.net/open/suricata/rules/botcc.rules`. Acessado em: 18/12/2017.

Erdem, O., Pektas, A., and Kara, M. (2018). Honeything: A new honeypot design for cpe devices. *KSII Transactions on Internet and Information Systems*, 12:4512–4526.

Gandhi, U. D., Kumar, P. M., Varatharajan, R., Manogaran, G., Sundarasekar, R., and Kadu, S. (2018). Hiotpot: Surveillance on iot devices against recent threats. *Wireless Personal Communications*, 103(2):1179–1194.

GARTNER (2017). 6.4 billion connected "things"will be in use in 2016. `https://www.gartner.com/newsroom/id/3165317`. Acessado em: 13/04/2017.

Hoepers, C., Steding-Jessen, K., and Chaves, M. H. P. C. (2007). Honeypots e honeynets: Definições e aplicações. `https://www.cert.br/docs/whitepapers/honeypots-honeynets/`. Acessado em: 18/12/2018.

Koniaris, I., Papadimitriou, G., Nicopolitidis, P., and Obaidat, M. (2014). Honeypots deployment for the analysis and visualization of malware activity and malicious connections. In *2014 IEEE International Conference on Communications (ICC)*, pages 1819–1824.

Marzano, A., Alexander, D., Fazzion, E., Fonseca, O., Cunha, Italo e Hoepers, C., Steding-Jessen, K., e Chaves, M. H. P. C., Guedes, D., and Meira Jr., W. (2018). Monitoramento e caracterização de botnets bashlite em dispositivos iot. *Simpósio Brasileiro de Redes de Computadores (SBRC)*, 36.

Pa, Y. M. P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., and Rossow, C. (2015). Iotpot: Analysing the rise of iot compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, Washington, D.C. USENIX Association.

Pathan, A.-S. K. (2014). *The State of the Art in Intrusion Prevention and Detection*. Auerbach Publications, Boston, MA, USA.

Simpson, A. K., Roesner, F., and Kohno, T. (2017). Securing vulnerable home iot devices with an in-hub security manager. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pages 551–556. IEEE.

Sochor, T. and Zuzcak, M. (2014). Study of internet threats and attack methods using honeypots and honeynets. In Kwiecień, A., Gaj, P., and Stera, P., editors, *Computer Networks*, pages 118–127, Cham. Springer International Publishing.

SPAMHAUS (2018). Spamhaus botnet threat report 2017. `https://www.spamhaus.org/news/article/772/spamhaus-botnet-threat-report-2017`. Acessado em: 05/03/2018.

II Workshop de Segurança Cibernética em Dispositivos Conectados
SBRC 2019
Sessão Técnica 2

Uma Avaliação de Algoritmos Criptográficos em Redes IEC 61850: Uma Abordagem Prática*

Rafael B. Scarselli[1], **Leonardo F. Soares**[1], **Igor M. Moraes**[1]

[1]Laboratório MidiaCom, PGC - TCC
Instituto de Computação – Universidade Federal Fluminense
Niterói – RJ – Brasil

{rscarselli,leonardofiorio}@id.uff.br, igor@ic.uff.br

Abstract. *This paper evaluates cryptographic algorithms applied to the GOOSE protocol in IEC 61850 communication networks for electrical substations. The IEC suggests the use of RSA for digital signature of GOOSE messages and at the same time defines a maximum communication latency of 3 ms for critical messages. Through practical experiments with devices with low computational power, the inviability of the RSA suggested by the IEC is confirmed. Results show that the AES symmetric cryptography algorithm with the CMAC technique meets the time constraints defined by IEC, even when the entire payload of a 459-byte packet is encrypted.*

Resumo. *Este artigo avalia algoritmos criptográficos aplicados ao protocolo GOOSE em redes de comunicação IEC 61850 para subestações de energia elétrica. A IEC sugere o uso do RSA para assinatura digital das mensagens GOOSE e ao mesmo tempo define uma latência máxima de comunicação de 3 ms para mensagens críticas. Através de experimentos práticos com dispositivos de baixo poder computacional, confirma-se a inviabilidade do RSA sugerido pela IEC. Os resultados mostram que o algoritmo de criptografia simétrica AES com a técnica CMAC atende às restrições de tempo definidas pela IEC, mesmo quando se cifra toda a carga útil de um pacote de 459 bytes.*

1. Introdução

As redes elétricas inteligentes (*smart grids*) estão em franca implantação em todo o mundo [Farhangi 2010]. Um Sistema Elétrico de Potência (SEP) [ABNT5460 1992] é a infraestrutura de geração, transmissão e distribuição de energia elétrica. Um SEP pode fazer parte de uma grande rede nacional e, por isso, o controle e gerenciamento é fundamental para manutenção do sistema. Uma parte fundamental de um SEP são as subestações de energia elétrica que estão sob o domínio de alguma concessionária de energia. É a partir das subestações que a energia é distribuída por bairros e regiões e chega às casas dos consumidores. O gerenciamento e controle das subestações é feito remotamente pela concessionária através de redes de comunicação privada e restritas aos administradores do sistema, sem qualquer interferência externa.

A norma IEC 61850 (*International Electrotechnical Commission*) define um padrão para comunicação de dispositivos eletrônicos inteligentes (IEDs, *Intelligent Electronic Device*) em subestações de energia elétrica. Os IEDs são relés digitais que permitem

*Este trabalho tem suporte financeiro da TAESA/ANEEL, CNPq, CAPES, FAPERJ e FAPESP.

implementação de lógicas para chaveamento, bloqueio e abertura. A comunicação entre os IEDs é responsável pela automação e controle de toda a subestação e também pelo gerenciamento feito pela concessionária. É através da rede de comunicação entre os IEDs que as falhas são contidas localmente na subestação, não permitindo que se espalhem em um efeito avalanche por toda a rede nacional de energia elétrica.

Os IEDs de uma subestação de energia se comunicam com outros IEDs através do protocolo de comunicação *Generic Object Oriented Substation Event* (GOOSE) [IEC61850 2003]. O GOOSE é implementado diretamente na camada de enlace da pilha de protocolos TCP/IP, pois suas mensagens transportam conteúdos genéricos e possuem requisitos estritos de tempo. Essa é a razão pela qual os IEDs enviam mensagens GOOSE. Para controlar os níveis desejados de tensão e corrente na subestação, os IEDs podem emitir comandos de controle como um disjuntor, por exemplo, caso detectem anomalias de corrente, tensão ou frequência oriundas dos sensores e equipamentos de energia que são conectados a eles. Estes equipamentos de padrão industrial são desenvolvidos com pequena capacidade de processamento, apenas o suficiente para suas funções básicas. Um IED comercial lançado em 2009 para redes IEC 61850 pela empresa ABB da série 670, por exemplo, possui processador IBM 3200PowerPC 750FX de 600 MHz [ABB 2009].

A norma IEC 61850 define um requisito de 3 ms para realização da comunicação entre IEDs com o protocolo GOOSE no requisito mais estrito, por isso ele é considerado um protocolo de tempo real *publisher/subscriber*. Isso justifica o fato do GOOSE ser implementado sobre a camada de enlace, diminui os tempos de geração, recebimento e processamento das mensagens, porém implica falta de segurança que poderia ser provida por camadas superiores da pilha TCP/IP. Isso se torna uma preocupação visto que não há controle de integridade e confidencialidade dos dados enviados através de mensagens GOOSE, facilitando ação de atacantes [Lopes et al. 2016]. As subestações tornaram-se mais conectadas às redes externas devido a novas demandas emergentes das *smart grids*, como por exemplo medidores inteligentes com comunicação bidirecional com a subestação, gerando potencial para atacantes externos à rede da concessionária.

A segurança da comunicação em tempo real de redes de automação de subestação de energia permanece um problema em aberto. Algoritmos criptográficos podem auxiliar na implementação segura do protocolo GOOSE. Porém, a aplicação desses algoritmos interfere nos tempos gastos para comunicação, pois são inseridos tempos de cifragem, decifragem e para cálculos de *hash*, por exemplo. Com a baixa capacidade de processamento dos dispositivos IEDs, a influência desses algoritmos pode ser ainda maior.

Este artigo tem o objetivo de avaliar algoritmos de criptografia aplicados ao protocolo GOOSE em uma bancada de testes experimental para verificar viabilidade da implementação dos algoritmos SHA256 (*Secure Hash Algorithm*), AES128 (*Advanced Encryption Standard*), RSA2048 (*Rivest-Shamir-Adleman*), CMAC (*Cipher-based Message Authentication Code*) e HMAC (*Hash MAC*) de acordo com o impacto nos tempos de comunicação com mensagens GOOSE encriptadas e o nível de segurança oferecido por cada algoritmo. Para isso, são implementados e executados os algoritmos de emissor e receptor de mensagens em Raspberry PI, configurados como um IED, a fim de comparar os resultados de tempo medidos com os requisitos estabelecidos pela norma IEC 61850 em um *hardware* com baixa capacidade computacional. Os resultados obtidos nos experimentos constatam a inviabilidade do uso do RSA recomendado pela norma IEC 62351.

Por outro lado, mostram que uma alternativa viável ao RSA é o algoritmo AES128 com segurança equivalente e desempenho de 475 μs de comunicação fim-a-fim, na técnica de código de autenticação, e 69 μs cifrando toda a carga útil de dados GOOSE. No caso do AES, há o desafio da distribuição de chaves que não é tratado neste artigo.

O restante desse artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados ao tema deste artigo, abordando principalmente o aspecto da segurança. A Seção 3 aborda as definições da norma IEC 61850 e 62351 e também traz informações de algoritmos criptográficos avaliados neste trabalho. A Seção 4 apresenta a metodologia empregada para avaliar experimentalmente os algoritmos criptográficos. Na Seção 5, a análise dos resultados é apresentada. Em seguida, na Seção 6 estão as conclusões deste artigo após os experimentos e também são propostos trabalhos futuros.

2. Trabalhos Relacionados

Existem diferentes trabalhos que abordam a segurança em redes de comunicação de subestações de energia [Wang and Lu 2013]. Um destes trabalhos [Hohlbaum et al. 2010] alerta que dispositivos de baixa capacidade de processamento, como os IEDs das subestações de energia, não são capazes de usar o algoritmo de chaves assimétricas RSA para assinar digitalmente mensagens GOOSE dentro dos requisitos de tempo da norma IEC 61850. Um IED comercial tipicamente possui um processador com relógio que vai de dezenas a poucas centenas de MHz de processamento [ABB 2009].

Novos estudos surgiram com propostas para atender aos requisitos de tempo da IEC 61850, desde então. Lu et. al. avaliam em um computador pessoal o tempo de processamento de algoritmos TESLA (*Timed Efficient Stream Loss-tolerant Authentication*) e HORS (*Hash to Obtain Random Subset*) junto com o RSA1024 [1] [Lu et al. 2012]. Nesta avaliação, varia-se o relógio da CPU do computador e apresenta-se qual o percentual de mensagens GOOSE transmitidas são entregues em menos de 3 ms. Os autores concluem que o RSA apresenta taxa de entrega inferior a 85% mesmo com uma CPU de 1,6 GHz.

Outro estudo [Yavuz 2014] propõe um novo esquema de autenticação *broadcast* com um protocolo de criptografia variante do RSA chamado RSA-condensado. A variante proposta é processada em menor tempo computacional, validada pelo experimento realizado em um computador portátil com processador Intel Core i7 de 1,6 GHz e 2 GB de memória RAM. O algoritmo RSA-condensado substitui uma operação de exponenciação por uma multiplicação, em comparação ao RSA. No cenário avaliado, não são especificados os detalhes das mensagens enviadas. Os resultados mostram um desempenho de 3792 μs para o RSA e de 226 μs para o RSA-condensado, considerando a média de tempo para enviar 10^4 mensagens. O trabalho comparou vários algoritmos (ECDSA, HORS, TV-HORS) ao RSA1024, todos considerando um nível de segurança de 80 bits[2]. Essa segurança é insuficiente para os dias atuais [Barker 2016] e foi testada em um *hardware* de maior capacidade que os IEDs.

Alguns estudos mostram uma nova abordagem para o problema. Como é economicamente inviável trocar todos os IEDs existentes nas subestações de energia, pois são equipamentos fabricados para muitos anos de longevidade, outra solução é utilização de

[1] RSA com chave de 1024 bits

[2] Comparação a um algoritmo simétrico com chave de 80 bits para ser quebrado através de força bruta

hardware de apoio. Uma pesquisa realizada em redes veiculares [Singla et al. 2015] desenvolveu um *hardware* especializado para criptografia utilizando GPUs (*Graphics Processing Unit*) com o algoritmo *Rapid Authentication* (RA) baseado no RSA Condensado [Yavuz 2014]. Utiliza-se uma placa NVidia Tesla K40c de 2880 cores de computação e 12 GB de memória RAM. O experimento foi executado em um computador pessoal com processador Intel Core i7 de 3,5 GHz e 16 GB de memória RAM. Também testa um SoC Tegra K1 com processador quadcore ARM Cortex A15 com 2,3 GHz e 192 núcleos de computação. O RSA2048[3] apresentou uma comunicação fim a fim de 4 ms. A GPU com o algoritmo RA processou em 0,21 ms e o SoC com o mesmo algoritmo teve desempenho de 2,6 ms. Na avaliação usou-se uma rajada de 8192 mensagens.

Outro trabalho [Miranda 2016] também usa um *hardware* adicional. O autor usa um NetFPGA para desenvolver um *firewall* de mensagens GOOSE aleatórias enviadas por um gerador de pacotes. O FPGA Virtex II Pro é utilizado com módulo GFIXED (*Message GOOSE Fixed*) e GPDU (*Protocol Data Unit GOOSE*) em *hardware*. Um módulo de criptografia (*Crypto*) também é utilizado. O *firewall* é testado com 10 mil mensagens do gerador de pacotes e 15 mil mensagens GOOSE de IEDs comerciais. Os algoritmos utilizados são o DES (*Data Encryption Standard*) para cifragem e o SHA3-512 para código de autenticação. O *firewall* implementado consegue filtrar com sucesso 100% das mensagens incorretas. Com o filtro DES foi registrado tempo de 77,39 μs e com o SHA-3 de 40,03 μs, considerando validação do quadro pelo firewall. O equipamento utilizado, no entanto, custa mais de $1.300,00 dólares [Digilent 2018].

A maioria dos trabalhos citados nesta seção utilizou computadores pessoais ou *hardware* específico para avaliar as propostas. A metodologia adotada neste artigo utilizará hardwares de Raspberry Pi 3. Além disso, será avaliado protocolo de cifra simétrica para substituir o RSA na assinatura digital e também na cifragem de toda a carga útil de dados da mensagem GOOSE, um cenário que não foi avaliado por outros autores.

3. Segurança na Comunicação em Redes IEC 61850

A implementação de algoritmos criptográficos em mensagens GOOSE é fundamental para evitar ataques cibernéticos em subestações de energia. Isso ocorre em virtude da implementação direta do protocolo GOOSE sobre a camada de enlace para atender aos requisitos estritos de tempo da norma IEC 61850. Um possível ataque ao protocolo GOOSE sem criptografia consiste na injeção de mensagens falsas com valores muito maiores que os vigentes no campo stNum [Kush et al. 2014], fazendo com que IEDs descartem mensagens legítimas com valores de stNum inferiores. As mensagens GOOSE legítimas com stNum menores do que o gerado pelo atacante são descartadas pelos nós receptores.

A *International Electrotechnical Commission* (IEC) estabeleceu a norma IEC 62351 [IEC62351 2007] com a definição de requisitos de segurança para vários protocolos, incluindo o GOOSE. Em particular, este artigo estuda as mensagens GOOSE de tempo real. Para essas mensagens, a norma IEC 62351 sugere o uso do RSA para assinatura digital das mensagens, garantindo autenticidade e integridade.

O protocolo GOOSE adota o modelo *publish/subscriber*, um nó publicador envia a mensagem e vários outros nós assinantes a recebem. Por isso, as mensagens GOOSE

[3]RSA com chave de 2048 bits.

são difundidas por *multicast*. A IEC 62351 estabelece que a mensagem GOOSE deve ser resumida com o algoritmo SHA256 e posteriormente esse resultado será assinado com o RSA. Isso garantirá a integridade da mensagem e a autenticidade. Entretanto, um estudo [Hohlbaum et al. 2010] mostra que o uso do RSA com chave de 1024 bits e com as restrições de tempo real exigidas pela norma eram possíveis em 2010 apenas com *criptochips* dedicados para o RSA. Ademais, o *National Institute of Standards and Technology* (NIST, Estados Unidos), entidade que estabelece padrões de segurança da informação, não recomenda mais o uso do RSA1024 [Barker 2016], sugerindo seu substituto o RSA2048 como requisito mínimo de segurança para futuras aplicações.

Permanece, então, a necessidade de alternativas ao RSA para proteger a comunicação entre IEDs em subestações através de mensagens GOOSE. A IEC 62351 indica que será adicionado um novo cabeçalho de segurança na mensagem, logo após o quadro GOOSE. No entanto, esta norma não especifica os detalhes de como isso será feito. Neste trabalho, os *bytes* extras resultantes do cabeçalho de segurança foram adicionados, assim como indica a norma, após o campo GOOSE PDU. Posteriormente, esses bytes são devidamente checados no destinatário da mensagem para garantir a entrega segura.

Os ataques às redes de subestação são devastadores [Hoyos et al. 2012]. O foco deste artigo é o ataque de falsificação no qual o atacante pode forjar uma mensagem GOOSE que pode causar grandes danos ao sistema elétrico em larga escala. Para evitar este ataque, dois requisitos de segurança são importantes de serem adicionados à comunicação entre IEDs: integridade e autenticidade. Desta forma, pacotes só são aceitos no destinatário se forem verificadas a integridade e a origem do pacote.

3.1. Algoritmos de Criptografia

A Tabela 1 enumera e resume as características dos algoritmos criptográficos usados na avaliação experimental desse trabalho.

O RSA [Rivest et al. 1978] é um algoritmo de criptografia assimétrica, isto é, seu funcionamento consiste na utilização de diferentes chaves para cifragem e decifragem de mensagens. Sua complexidade é baseada no problema de fatoração da multiplicação de números primos grandes, o que torna improvável a tentativa de descoberta das chaves utilizadas. Com o RSA é possível prover autenticidade, ou seja, ter a garantia de que uma mensagem foi enviada por determinado emissor. Para tanto, um emissor pode enviar junto da mensagem a sua assinatura. A assinatura digital é o processo em que o emissor da mensagem gera um resumo desta mensagem e cifra esse resumo com sua chave privada RSA. Essa assinatura digital é exclusiva desse emissor e da respectiva mensagem, e é enviada juntamente com a mensagem. O destinatário, ao receber a mensagem e a assinatura, faz a conferência dessa assinatura digital com a chave pública RSA do emissor. Se a conferência for positiva, consegue-se ter a garantia do emissor e da integridade da mensagem.

O algoritmo SHA256 [Mendel et al. 2006] é uma função *hash* que consiste em um conjunto de cálculos efetuados por um algoritmo para geração de uma saída de tamanho único dada uma entrada de tamanho qualquer. O resultado de uma função *hash* derivado do conteúdo de uma mensagem pode ser anexado a mensagem pelo emissor e enviada ao seu destino. Ao receber, o destinatário aplica a função *hash* ao conteúdo recebido e compara o resultado com o *hash* contido na mensagem. Se forem iguais, o conteúdo

está íntegro, caso contrário, houve algum tipo de alteração durante a comunicação. A aplicação deste algoritmo seguido do RSA proporciona integridade e autenticidade da comunicação e é denominada assinatura digital.

Tabela 1. Algoritmos de segurança e seus parâmetros

Algoritmo	Tamanho da Chave (bits)	Tamanho do Bloco (bits)	Tamanho da Saída (bytes)	Nível de Segurança [Barker 2016]
Nenhum	0	0	Igual entrada	Nenhuma
SHA256	0	512	32	128 bits
AES128	128	128	Igual entrada	128 bits
RSA2048	2048	Igual entrada	Igual entrada	112 bits
HMAC SHA256	128	512	32	256 bits
CMAC AES128	128	128	16	128 bits

Uma alternativa avaliada neste trabalho é a utilização da função SHA256 combinado com o algoritmo de chave simétrica denominado AES128 substituindo o RSA sugerido na norma IEC 62351. O AES utiliza a mesma chave para cifragem e decifragem de conteúdos e implementa cifragem de bloco. A cifragem de bloco opera recebendo como entrada um conjunto de bytes do conteúdo a ser encriptado a cada iteração. Através da Tabela 1 é possível observar que o AES128[4] provê maior nível de segurança [Barker 2016] quando comparado ao RSA2048 e, por esse motivo, é uma solução candidata a ser implementada para prover segurança a quadros GOOSE substituindo o RSA na assinatura digital das mensagens.

Outro mecanismo usado para prover autenticidade e integridade é o *Message Authentication Code* (MAC). O HMAC (*Hash MAC*) é um algoritmo que implementa esta estrutura utilizando uma função *hash* junta a uma chave criptográfica. Ele é composto de duas etapas. Durante a sua execução são gerados dois componentes: o *keypad* interno (i_keypad) e o *keypad* externo (o_keypad). A primeira etapa é a geração de um *hash* do resultado da operação XOR da mensagem com o i_keypad. Na segunda etapa, é feito um *hash* da aplicação do XOR entre o resultado da etapa anterior com o o_keypad.

Neste trabalho, o HMAC é implementado com a função *hash* SHA256 devido a sua utilização na assinatura digital do RSA mencionado anteriormente. Cada mensagem gera uma saída diferente no HMAC que é concatenada ao final da mensagem em texto plano. Já no destino, o receptor consegue repetir a aplicação do HMAC à mensagem em texto plano e comparar com o conteúdo cifrado que foi anexado a mensagem pelo emissor.

O último algoritmo implementado é o CMAC (*Cipher MAC*) AES128. Assim como o anterior, também é possível prover autenticidade e integridade, porém o algoritmo usado é de cifragem de bloco ao invés da função *hash*. A implementação é feita de acordo com a RFC4493 que passou por rotinas de testes da própria RFC e do NIST. O algoritmo recebe blocos da mensagem a ser cifrada e, à medida que os *n* blocos são cifrados pelo algoritmo AES, o algoritmo recebe como entrada também o resultado da cifragem do bloco *n-1*. Portanto, a saída deste algoritmo tem um tamanho fixo.

[4]AES com chave de 128 bits.

Além dos algoritmos MAC acima implementados, também é realizado o teste de cifragem total da mensagem utilizando o RSA2048 e o AES128. Na Seção 5 isso é abordado com maiores detalhes.

4. Metodologia de Avaliação

Para avaliar os algoritmos criptográficos, montou-se uma bancada de testes com objetivo de se aproximar o máximo possível de uma rede de subestação composta por alguns IEDs. O *hardware* escolhido foi o Raspberry Pi 3 Model B. Este equipamento possui desempenho semelhante a dos IEDs: arquitetura ARM, baixo *clock* de CPU, pequena quantidade de memória RAM, armazenamento limitado, interface de rede 10/100 Mb/s e funcionamento sem resfriamento ativo [ABB 2009]. O modelo escolhido possui um processador de 1.2 GHz Broadcom BCM2837 64 bits e 1 GB de memória RAM, de baixo custo, em torno de 30 dólares americanos. Os trabalhos anteriores utilizaram computadores pessoais de arquitetura x86 e processadores mais poderosos.

Cada dispositivo é instalado com o sistema operacional Ubuntu Mate 16.04.02, compilador GCC 4.7.4, biblioteca Libgcrypt20 e suas dependências. Todos os algoritmos de criptografia usados neste trabalho, são implementações próprias validados com vetores de teste do NIST. A única exceção é o algoritmo RSA, para o qual é utilizada uma implementação da biblioteca Libgcrypt [Project 2018], apenas para comparação com os algoritmos propostos visto que outros trabalhos já identificaram a inviabilidade do RSA.A implementação própria dos algoritmos apresenta melhor desempenho comparado aos de bibliotecas como a Libgcrypt, pois possuem código mais enxuto e objetivo, com menos etapas de processamento.

Quatro dispositivos são conectados para representar nós de uma subestação se comunicando através do protocolo GOOSE por uma rede Ethernet. É utilizado um comutador Ethernet Encore ENH908-NWY de 8 portas conectando todos os nós através de cabos de rede Cat6. Além disso, é conectado à rede um computador pessoal apenas para captura de pacotes, não influenciando o tráfego durante os testes.

A Figura 1 é uma captura de tela do *software* farejador Wireshark. Ela mostra um quadro GOOSE como definido na norma IEC 61850 e capturado durante os experimentos. As informações do quadro Ethernet são padronizadas em todas as mensagens, bem como todas as informações da carga útil IEC 61850, com uma única exceção. Dentro da carga útil, existe um campo chamado GOOSE.t que marca uma estampilha de tempo. Em nosso experimento, em todos os pacotes enviados, o único campo que sempre é alterado é o GOOSE.t que sempre recebe uma nova marcação de tempo para cada pacote gerado em tempo de execução. Portanto, o *hash* de cada pacote é substancialmente diferente um do outro.

Os experimentos consistem em medições de tempo das execuções dos algoritmos com diferentes soluções de segurança implementadas. Os programas implementados na linguagem C possuem marcadores que registram o tempo de processamento total de cada execução e divide esse tempo pela quantidade de quadros gerados, exibindo a média de tempo de cada teste ao término de cada execução. Como todos os quadros são diferentes, por exemplo, em uma rajada de 1000 pacotes, nenhum é replicado, tornando o processo de cifragem diferente em cada pacote gerado.

Para a avaliação de algoritmos criptográficos, são considerados cinco opções:

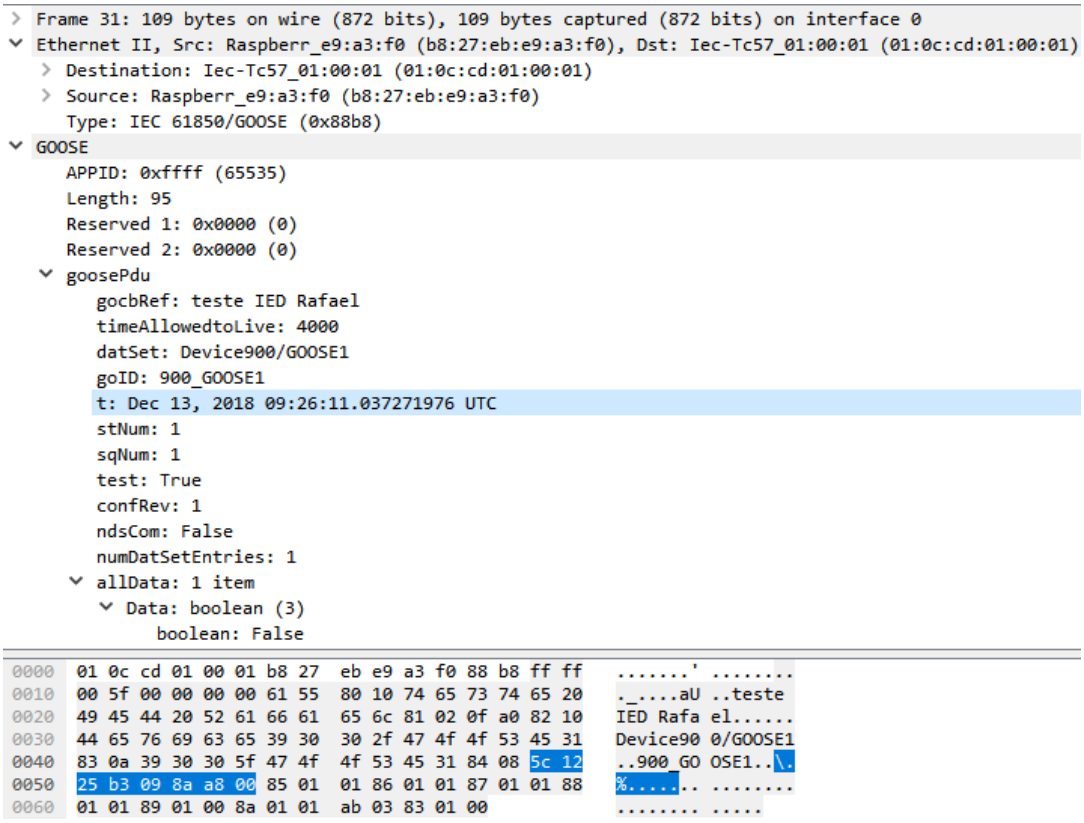


Figura 1. Estrutura do pacote GOOSE padrão utilizado nos testes. Apenas o campo GOOSE.t é alterado em cada pacote novo, em destaque

SHA256, AES128, RSA2048, HMAC SHA256 e CMAC AES 128. Todos os algoritmos são selecionados considerando as recomendações mínimas de segurança do NIST [Barker 2016] para implementações atuais e futuras. A Tabela 1 mostra em detalhes os algoritmos e suas características consideradas neste experimento. Para ter uma linha de base, todos os testes também são realizados sem nenhuma configuração de segurança, para que seja feita uma comparação direta da opção sem segurança para as outras alternativas avaliadas.

A avaliação dos tempos de geração dos quadros GOOSE é feita testando rajadas de 20, 100, 500, 1000 e 5000 mensagens. Com essas quantidades é possível observar o comportamento das médias calculadas para cada algoritmo de criptografia à medida que o número de mensagens da rajada aumenta. Em cada tipo de teste, são feitas 30 amostras de execução do algoritmo. Os gráficos da Seção 5 incluem os intervalos de confiança para um nível de confidencialidade de 95% que, em alguns casos, são valores muito pequenos e não ficam visíveis na escala utilizada.

Considerando todas as opções de algoritmos de segurança, mais as variações de tamanhos de rajadas e a quantidade de amostras por teste, ainda foram realizados 4 tipos diferentes de teste. O primeiro é a marcação de tempo no envio da mensagem, somente no transmissor. O segundo é a marcação de tempo na recepção da mensagem, nos receptores. Nesse caso foram utilizados três nós receptores e calculada uma média entre os tempos de cada um. O terceiro teste faz uma variação do tamanho do quadro GOOSE, adicionando preenchimento no mesmo, para analisar o comportamento da variação de tempo

nos algoritmos em relação ao tamanho do pacote. Foram adicionados 50, 100, 200, 350 bytes ao pacote padrão para verificar o comportamento dos algoritmos com uma carga útil maior. O quarto e último teste é realizado comparando dois algoritmos de criptografia cifrando todo a carga útil da mensagem, garantindo assim a confidencialidade, requisito de segurança além do que a norma exige, mas que é desejável.

A definição desta metodologia, em primeiro lugar, é a escolha do Raspberry por ser um *hardware* que se assemelha a um IED de produção em suas características. É estabelecido que os testes utilizem o algoritmo recomendado pela norma, para validar informações de que ele é inviável [Hohlbaum et al. 2010, Lu et al. 2012, Hoyos et al. 2012]. Posteriormente são avaliadas alternativas que proporcionem integridade às mensagens. É avaliado o AES em conjunto com o SHA256 para gerar um *hash* cifrado da mensagem e proporcionar integridade na comunicação. Por fim, as variantes MAC são testadas usando o *hash* SHA256 e a cifra simétrica AES128.

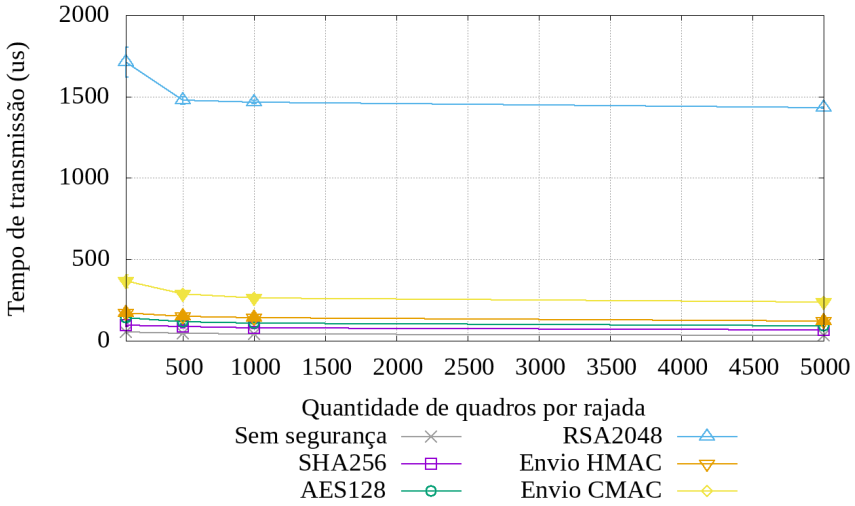
5. Avaliação dos Resultados

Como característica das redes IEC 61850, o modelo de comunicação *publisher / subscriber* apresenta um nó publicador que envia as informações e um ou vários nós assinantes, que são receptores da mesma mensagem enviada via *multicast*. Foram realizadas medições separadas da transmissão e recepção devido à diferença de relógio dos diferentes Raspberry. O *Network Time Protocol* foi testado, porém descartado pois apresentou uma precisão de tempo que afetava os resultados em vários microssegundos.

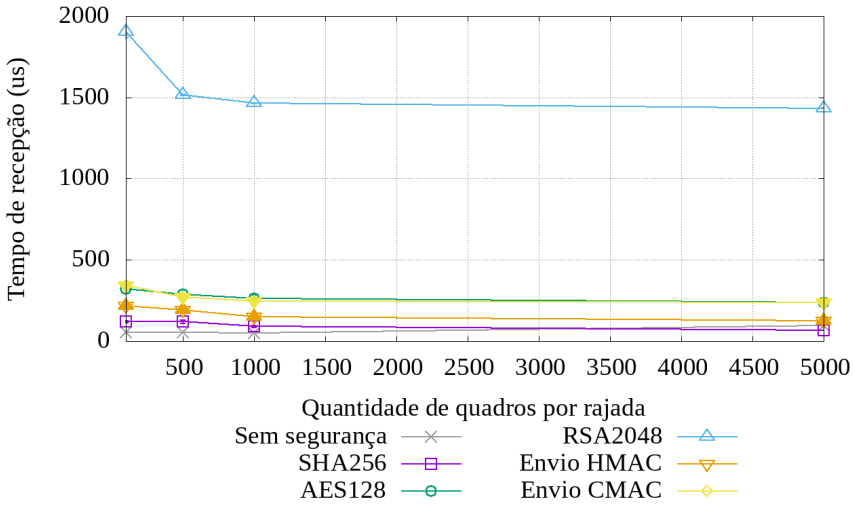
O tempo médio de envio de pacotes GOOSE em função do tamanho da rajada é apresentado na Figura 2a. Observa-se que quanto maior o tamanho da rajada, menor é o tempo de transmissão por mensagem em todos os algoritmos criptográficos avaliados. Esse comportamento ocorre porque as 10 primeiras mensagens da rajada sempre levam mais tempo de envio devido à abertura do *socket* no sistema operacional. Sobre os algoritmos criptográficos é possível observar que todos processam e enviam pacotes em menos de 500 μs, com exceção do RSA que opera próximo dos 1500 μs. O pacote com nenhuma segurança, ou seja, mensagem GOOSE padrão, tem uma média de tempo de 33 μs. Como o SHA256 marca a média de 66 μs, presume-se o tempo de calcular o *hash* é de 33 μs. Assim, todos os algoritmos, menos o RSA, operam abaixo dos 250 μs para envio incluindo o tempo de *hash*.

O tempo médio de recepção das mensagens GOOSE em função do tamanho da rajada é apresentado na Figura 2b. Observa-se que o tempo médio de recepção diminui com o aumento do tamanho da rajada, assim como aconteceu no tempo de transmissão. Observa-se também que cada um dos algoritmos criptográficos avaliados possuem tempos similares para processamento e transmissão das mensagens e recepção e processamento das mensagens. A exceção é o AES128 que apresentou tempo maior comparado ao envio, uma vez que a compilação para arquitetura ARM gera mais ciclos de relógio no processo de decifragem do AES [Bertoni et al. 2002]. Como observado por outros autores, o RSA se torna inviável para dispositivos de baixo processamento, pois somente a soma dos tempos de envio e recepção atinge o limite de 3ms definido pela norma IEC 61850, sem considerar o tempo de propagação que não foi medido nesses experimentos.

Os tempos médios de transmissão e recepção das mensagens GOOSE para uma rajada de 5000 pacotes podem ser observados na Tabela 2. Na metodologia de teste apre-



(a) Transmissão GOOSE



(b) Recepção GOOSE

Figura 2. Tempos de transmissão/recepção das mensagens GOOSE em função do tamanho da rajada para cada algoritmo criptográfico.

sentada nesta tabela, um nó transmissor envia mensagens para outros três nós receptores. Os valores na tabela são a média dos tempos de recepção das mensagens recebidas pelos 3 nós. Os resultados apresentados de tempo de comunicação são a soma dos resultados de envio e recepção, desconsiderando o tempo de propagação na rede.

Todos os algoritmos de segurança se mostram viáveis de implementação em aplicação real nas mesmas condições do cenário proposto com uma observação. O RSA está no limite do tempo de 3 ms exigido pela norma. Como os valores registrados mostram a média, é possível concluir que o RSA se torna inviável, pois apresenta o risco de extrapolar os 3 ms. Outras variáveis podem aumentar o tempo registrado como concorrência do processador do nó, condições de rede, aumento de tamanho da mensagem GOOSE, etc.

A Figura 3 apresenta o tempo médio de transmissão de mensagens GOOSE em função da quantidade de bytes acrescentados a cada mensagem originalmente com ta-

Tabela 2. Valores da média das amostras em μs.

	Nenhuma Segurança	SHA256	AES128	RSA2048	HMAC SHA256	CMAC AES128
Transmissão	33	66	93	1467	120	236
Desvio Padrão	2,56	3,37	2,59	10,74	2,58	3,66
Recepção	96	67	237	1434	124	239
Desvio Padrão	1,27	0,93	3,97	1,40	0,40	0,53
Soma	129	133	330	2901	244	475
Pior Tempo	157	140	339	2932	262	505

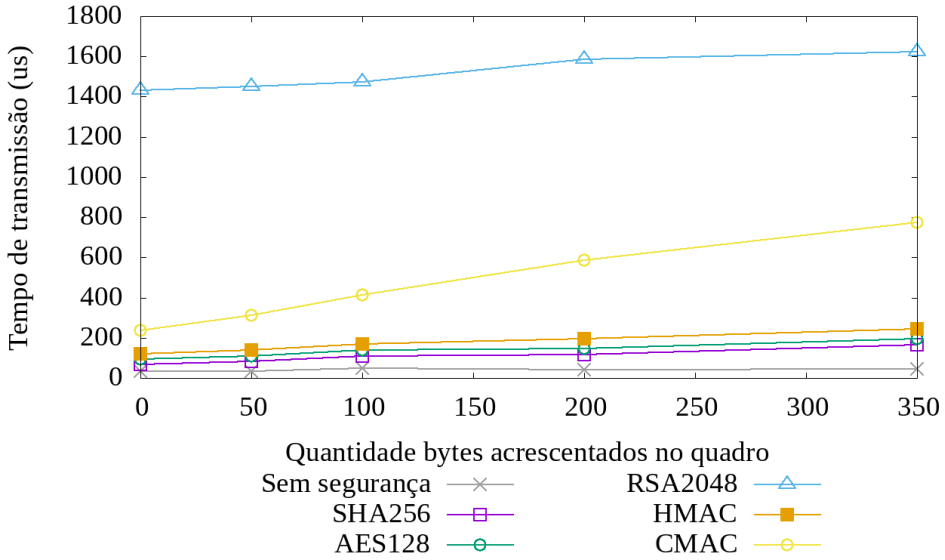


Figura 3. Tempo médio de transmissão das mensagens GOOSE em função da quantidade de bytes acrescentados a um quadro de 109 bytes.

manho de 109 bytes e rajada de 5000 mensagens. O algoritmo RSA apresenta tempo de processamento de envio superior a 1400 μs, comprovando resultados obtidos em outros trabalhos [Hohlbaum et al. 2010][Yavuz 2014]. Os tempos obtidos para o CMAC mostram que esse algoritmo tem resultados mais influenciados pelo aumento da quantidade de bytes do quadro a ser cifrado devido à característica de cifragem de blocos em cadeia mais complexa. Mesmo assim, os tempos para o quadro de maior tamanho do experimento (459 bytes) não ultrapassam 800 μs. Os demais algoritmos de criptografia apresentam crescimento linear, sendo menos influenciado pelo crescimento do tamanho do quadro a ser enviado do que o CMAC.

Os algoritmos SHA256, AES128 e HMAC e a configuração sem segurança, mostram um crescimento menor em função do aumento no tamanho da mensagem. Observa-se que em um comportamento normal, no qual a mensagem GOOSE vai variar entre 150 a 250 bytes, essas quatro variantes operam com tempo de transmissão abaixo dos 200 μs.

Como já mencionado, os algoritmos AES128 e RSA2048 receberam como entrada um resultado da função *hash* SHA256 nos experimentos. Para uma avaliação mais

detalhada de como se pode usar os algoritmos criptográficos, é realizado um último teste de desempenho. O objetivo é analisar se é possível cifrar toda a carga útil da mensagem GOOSE para prover confidencialidade. A Figura 4 apresenta um gráfico deste teste.

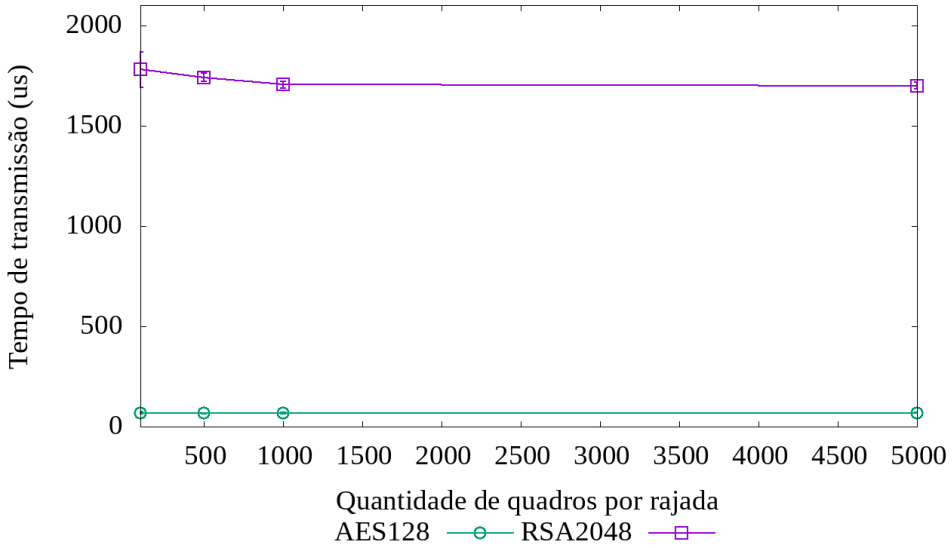


Figura 4. Tempo de cifragem e envio da mensagem GOOSE com carga útil totalmente cifrada em função da quantidade de pacotes enviados.

Observa-se uma variação mínima do tempo de transmissão em função da quantidade de pacotes enviados. O RSA apresentou desempenho em torno de 1700 μs para processar o envio com todo a carga útil cifrada. Comparando com a Figura 2a, é possível notar que houve um aumento do tempo de transmissão 1467 para 1700. Como o tamanho do quadro era o mesmo, o que mudou foi a quantidade de bytes cifrados pelo RSA. Por sua vez o AES128 enviou o quadro em tempo próximo de 69 μs, independente da quantidade de pacotes enviados na rede. Esse desempenho é melhor do que o observado na Tabela 2 para o mesmo algoritmo. Isso demonstra que o AES cifrando uma carga útil de 95 bytes (carga útil do quadro de 109 bytes padrão menos os 14 bytes do quadro Ethernet) é executado em menos tempo do que calcular o *hash* com SHA256 do quadro inteiro de 109 bytes e posteriormente cifrar com o AES os 32 bytes resultantes, procedimento este feito no primeiro teste.

6. Conclusão e Trabalhos Futuros

Neste artigo foi realizada uma avaliação experimental de algoritmos criptográficos candidatos à implementação de segurança no protocolo GOOSE. Deseja-se que estes algoritmos sejam capazes de garantir requisitos como integridade e autenticidade sem que os requisitos de tempo de entrega de mensagem da norma IEC 61850 sejam extrapolados.

Os resultados mostram que algoritmo AES, leva somente 12% do tempo para enviar uma mensagem GOOSE segura em relação ao tempo para enviar a mesma mensagem com o RSA. A comunicação fim-a-fim com o AES fica em torno de 330 μs, muito abaixo dos 3ms exigidos pela norma, tornando possível sua implementação em *hardware* ainda mais restrito do que o Raspberry. O AES também mostra que é possível cifrar totalmente

a carga útil das mensagens GOOSE e manter tempo de envio em 69 μs para um quadro de 109 bytes, provendo a confidencialidade, requisito acima do exigido pela norma.

Finalmente, outros dois algoritmos se mostram promissores e viáveis. Os algoritmos de MAC proporcionam a autenticidade da mensagem. O HMAC baseado em SHA256 apresenta tempo de comunicação fim-a-fim de 244 μs e há pouca variação neste tempo ao se aumentar o tamanho das mensagens enviadas. O CMAC implementado com o algoritmo AES128 tem uma variação maior com o aumento do tamanho das mensagens, mas ainda apresentou um tempo médio de transmissão abaixo de 800 μs para um pacote de 459 bytes, se mostrando também uma alternativa de aplicação em redes IEC 61850 com GOOSE. O CMAC é mais indicado para cenários com maiores restrições de memória nos nós, pois utiliza carga extra de segurança 50% menor que os outros algoritmos.

Os experimentos descritos abordam testes com diferentes algoritmos em um mesmo *hardware* em configuração padrão, implicando que as CPUs utilizadas possuam a mesma frequência de relógio. Um trabalho futuro pode inserir a frequência do processador como uma nova variável permitindo estudo dos algoritmos também em função da frequência do relógio do processador. Isso possibilitará avaliações direcionadas a requisitos de *hardware* menores ainda, atendendo a IEDs mais antigos em uso nas subestações.

Um dos pontos abertos desta pesquisa é a forma de distribuição das chaves simétricas na rede. Há uma corrente de estudos de implementações de SDN (*Software Defined Networking*) nas redes IEC61850 e esta é uma das possíveis maneiras de realizar a distribuição segura de chaves simétricas na rede e fazer a gestão de troca de chaves ao longo do tempo. Além disso, a aplicação do MAC no texto em claro bem como opções do AES de cifragem autenticada serão estudadas com maior cautela em trabalhos futuros.

Foi utilizado *hardware* similar aos IEDs devido a impossibilidade de utilizar um equipamento real em virtude de patentes e *softwares* proprietários dos fabricantes. Outro objetivo futuro é buscar parcerias com fabricantes de *hardware* IED para implementar os algoritmos diretamente no *firmware* desses equipamentos e realizar novos testes.

Referências

ABB, G. (2009). Manual técnico dos ieds da série 670. `https://library.e.abb.com/public/9830608e2e48f75fc12576f10031debf/1MRK580172-XEN_A_en_670_series_self_supervision.pdf`. Último acesso março de 2019.

ABNT5460 (1992). Sistemas elétricos de potência. In *ABNT NBR 5460:1992*. Associação Brasileira de Normas Técnicas. https://www.abntcatalogo.com.br/norma.aspx?ID=4123.

Barker, E. (2016). NIST special publication 800-57 part 1, revision 4, recomendation for key management. In *NIST Special Publication 800-57 Part 1 Revision 4*, page 160. National Institute of Standards and Technology.

Bertoni, G., Breveglieri, L., Fragneto, P., Macchetti, M., and Marchesin, S. (2002). Efficient software implementation of aes on 32-bit platforms. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 159–171. Springer.

Digilent (2018). Website do NetFPGA Virtex-II Pro FPGA Development System. `https://store.digilentinc.com/`

netfpga-virtex-ii-pro-fpga-development-system/. Último acesso em dezembro de 2018.

Farhangi, H. (2010). The path of the smart grid. *IEEE Power and Energy Magazine*, 8(1):18–28.

Hohlbaum, F., Braendle, M., and Alvarez, F. (2010). Cyber security practical considerations for implementing IEC 62351. In *PAC World Conference*.

Hoyos, J., Dehus, M., and Brown, T. X. (2012). Exploiting the GOOSE protocol: A practical attack on cyber-infrastructure. *2012 IEEE Globecom Workshops*, pages 1508–1513.

IEC61850 (2003). Communication networks and systems in substations. In *IEC 61850 Standard*. International Electrotechnical Commission.

IEC62351 (2007). Power systems management and associated information exchange - data and communications security. In *IEC 62351 Standard*. International Electrotechnical Commission.

Kush, N., Ahmed, E., Branagan, M., and Foo, E. (2014). Poisoned GOOSE: Exploiting the GOOSE Protocol. In *Proceedings of the Twelfth Australasian Information Security Conference - Volume 149*, AISC '14, pages 17–22.

Lopes, Y., Fernandes, N., Castro, T., and Muchaluat-Saade, V. (2016). Desafios de segurança e confiabilidade na comunicação para smart grids. *XVI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*.

Lu, X., Wang, W., and Ma, J. (2012). Authentication and integrity in the smart grid: An empirical study in substation automation systems. *International Journal of Distributed Sensor Networks*, 2012.

Mendel, F., Pramstaller, N., Rechberger, C., and Rijmen, V. (2006). Analysis of step-reduced SHA-256. In *International workshop on fast software encryption*, pages 126 – 143.

Miranda, J. C. (2016). *Segurança Cibernética com Hardware Reconfigurável em Subestações de Energia Elétrica Utilizando o Padrão IEC 61850*. PhD thesis, Universidade de São Paulo, Escola de Engenharia de São Carlos.

Project, T. G. (2018). Libgcrypt home page. https://www.gnupg.org/software/libgcrypt/index.html. Ultimo acesso em dezembro de 2018.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120 – 126.

Singla, A., Mudgerikar, A., Papapanagiotou, I., and Yavuz, A. A. (2015). HAA: Hardware-accelerated authentication for internet of things in mission critical vehicular networks. In *Milcom*, pages 1298–1304.

Wang, W. and Lu, Z. (2013). Cyber security in the smart grid: Survey and challenges. *Computer Networks*, 57(5):1344 – 1371.

Yavuz, A. A. (2014). An efficient real-time broadcast authentication scheme for command and control messages. *IEEE Transactions on Information Forensics and Security*, 9(10):1733–1742.

O impacto das fragilidades da comunicação 802.11 em competições de robótica

Júlio C. M. de Resende[1], André F. J. Santos[1], Thiago S. Gomides[1], Massilon Lourenço[1], Flávio L. Schiavoni[1], Marcos A. M. Laia[1]

[1]Departamento de Ciência da Computação
Universidade Federal de São João Del Rei (UFSJ), MG – Brasil

{julio.cmdr,andrefelipes,gomides,massilon,fls,marcoslaia}@ufsj.edu.br

Abstract. *The robot competitions had served as a scenarios for the study of several areas on computer science field. In these competitions, the quest for real-time control can have a side effect that is little concern for network communication security and performance. In this paper, a communication breakdown identified in a tournament pointing out the possible causes of the problem. This vulnerability scenario can be extended to allow a broader discussion involving the communication of embedded systems with the 802.11 protocol.*

Resumo. *As competições de robôs têm servido de cenário para o estudo de diversas áreas da ciência da computação. Nestas competições, a busca por um controle em tempo real pode trazer um efeito colateral que é a pouca preocupação com a segurança e o desempenho da comunicação em redes. Neste artigo, apresentamos uma falha de comunicação identificada em um torneio apontando as possíveis causas do problema. Este cenário de vulnerabilidade pode ser ampliado para permitir uma discussão mais ampla envolvendo comunicação de sistemas embarcados pelo protocolo 802.11.*

1. Introdução

Atualmente, diversas as aplicações utilizam os sistemas embarcados, como, por exemplo, os segmentos da robótica e o futebol de robôs. Esta modalidade esportiva, apresentada na Seção 2, atua de maneira interdisciplinar envolvendo conceitos computacionais em conjunto com outras áreas de conhecimento como visão computacional, inteligência artificial, programação de sistemas, robótica, redes de computadores e sistemas distribuídos. Na área de estudo de redes de computadores, o futebol de robôs tem motivado as pesquisas sobre a sincronização e controle da comunicação de sistemas em tempo real.

No Brasil, a *Latin American and Brazilian Robotics Competition* (LARC) é exemplo de torneio da comunidade nacional de robótica. Nossa equipe, a UaiSoccer VSS da Universidade Federal de São João del-Rei, teve a oportunidade de participar do LARC–XVII (2018), conforme apresentado na na Seção 2.1. Durante a competição, foram identificados problemas com a comunicação sem fio e este trabalho motiva-se na discussão deste tema.

Motivados na compreensão das falhas de comunicação enfrentadas e na busca por possíveis soluções, este trabalho apresenta uma discussão sobre a segurança das redes sem fio em competições de robótica. Os ataques de comunicação ao meio sem fio nestes

eventos são proibidos e, consequentemente, a preocupação com segurança é ignorada. Em contrapartida, devido a ausência dos registros da comunicação sem fio durante o evento, a tarefa de analisar as falhas de comunicação ocorridas ou mesmo recriar o ambiente é de extrema complexidade. No entanto, é possível descrever algumas circunstâncias que poderiam ter colaborado para possíveis vulnerabilidades. Ainda assim, devido à sobrecarga da rede, acreditamos que as falhas enfrentadas pela equipe UaiSoccer VSS podem ser resultado de causas naturais inerentes ao espectro de problemas enfrentados na área de segurança de redes de computadores. Por outro lado, este cenário nos permite discutir sobre ataques hipotéticos que poderiam ser realizados, conforme apresentado na Seção 3.

Na Seção 4, discutimos a segurança dos sistemas embarcados com enfoque em competições de robótica. Apesar da unicidade destes eventos, acreditamos que as circunstâncias aqui relatadas podem ser aplicadas para outras situações de controle de dispositivos embarcados que utilizam fortemente a sincronização em redes. Por fim, trazemos na Seção 5 algumas considerações finais e trabalhos futuros.

2. Futebol de Robôs

Entre as aplicações em que o uso de sistemas embarcados se faz necessário, estão a cooperação e competição de robôs móveis. Nesta área, destaca-se o futebol de robôs: um ambiente de topologia dinâmica e de tempo real que engloba conhecimentos de áreas como visão computacional, aprendizado de máquina, controle e redes de computadores [Buttazzo 2011].

Esse tipo de sistema demanda uma comunicação de dados de extrema confiabilidade, que são gerados a partir de eventos harmônicos ao estado atual do sistema. Atualmente, a competição de futebol de robôs está dividida em várias categorias que são praticadas em todo mundo. Entre as diversas competições existentes no mundo, destaca-se a LARC (*Latin American and Brazilian Robotics Competition*), um evento de robótica que ocorre uma vez ao ano em algum país da América do Sul, habitualmente o Brasil. A LARC abrange entusiastas da robótica que vão desde estudantes do ensino fundamental até doutores e pós doutores. Além do futebol, dentro do evento são praticadas categorias de robótica em que robôs autônomos realizam atividades como resgate de vítimas em ambientes desastrosos, auxílio doméstico, dança, corrida, dentre outras. Na LARC, as competições de futebol de robôs são divididas em categorias como simulação, robôs humanoides e robôs que utilizam rodas. Dentre todas, a categoria *IEEE Very Small Size Soccer* (IEEE VSSS) [IEEE 2009] se destaca por utilizar robôs que usam rodas de tamanho reduzido e controlados remotamente por um computador.

Nesta categoria, cada equipe tem direito a colocar até três robôs em campo. Uma câmera suspensa sobre o campo atua como entrada do sistema e a informação que detalha qual ação a ser tomada é enviada para os robôs a cada instante durante o jogo. Por isto, as vulnerabilidades na comunicação entre o computador e os robôs configuram um sistema distribuído de tempo real em que qualquer falha de comunicação pode resultar no não funcionamento do sistema.

2.1. A equipe UaiSoccer VSS e sua participação na LARC 2018

A UaiSoccer VSS é uma equipe de robótica que compete na categoria de futebol de robôs IEEE Very Small Size Soccer. O projeto teve inicio no ano de 2016 por uma iniciativa de membros do Departamento de Ciência da Computação (DCOMP) da UFSJ, que tinham

como objetivo o desenvolvimento e a aplicação de algoritmos de navegação em robôs reais.

A proposta da equipe para o LARC 2018, realizado em João Pessoa - PB, foi utilizar uma arquitetura de software separada em módulos, permitindo assim fácil manutenção. O primeiro módulo, de Visão Computacional, foi responsável por adquirir as imagens de uma câmera conectada ao computador via USB e processar estas imagens identificando a posição e a angulação de todos os robôs presentes no campo e da bola. Este módulo processa 60 fps, e a cada quadro processado uma cadeia de caracteres contendo todas as informações retidas do ambiente é enviada via socket UDP para o próximo módulo, que é executado no mesmo computador.

O segundo módulo, de navegação e controle, implementa estratégias de jogo, gera trajetórias para os robôs e envia comandos de velocidades para as duas rodas de cada robô. Cada comando é enviado para o seu respectivo robô 60 vezes por segundo, através de um socket UDP. Os robôs possuem um microcontrolador ESP8266, que conta com uma antena WiFi 2.4GHz acoplada para o recebimento dos comandos. O processamento do robô é dividido entre receber um pacote e executar a ação correspondente.

Esta necessidade de comunicação em tempo real para o controle dos robôs se mostrou uma estratégia passível de falhas durante a participação da equipe UaiSoccer VSS no LARC 2018. Muitas vezes, esta comunicação foi comprometida e impediu o funcionamento correto dos robôs. Entendendo que tal dificuldade de comunicação estava ocorrendo por falhas na rede, a UaiSoccer VSS optou por jogar com apenas um dos três robôs nos horários de maior movimentação no evento pois assim havia uma redução no número de pacotes trafegando pela rede, e consequentemente melhor desempenho na partida. Nos jogos e treinos que ocorreram na parte da noite, quando muitos competidores já haviam deixado o evento, foi possível colocar os 3 robôs para jogar e apresentar um desempenho próximo ao obtido em laboratório sem interferência externa, o que possibilitou vencer algumas partidas. O problema apresentado por nossa equipe também foi notado por outras equipes que participavam em outras modalidades, como as que envolviam Drones e/ou robôs de maior porte.

3. Exploração das Vulnerabilidades

Em competições de robótica, a exploração das vulnerabilidades dos protocolos de comunicação ou ataques destinados a eles é proibida. Por outro lado, o comitê organizador destes eventos não oferece mecanismos que permitem identificação ou defesa desta prática. Portanto, esta seção considera algumas das vulnerabilidades passíveis de exploração a partir dos ataques de comunicação. Neste contexto, os ataques destinados a interferências de sinais, como os *Jamming Attacks*, poderiam ser responsáveis pela sobrecarga do canal sem fio e, dessa forma, causariam falhas de comunicação entre robôs e servidores. Assim, nos ataques de interferência de sinal, um atacante busca proporcionar instabilidades nos dispositivos de rede baseados no protocolo 802.11. A partir de um conjunto de ferramentas, o atacante pode emitir também ruídos nas frequências de operação do protocolo e, com isso, interromper a comunicação entre os dispositivos. Além disso, os ataques de *Jamming* são de grande abrangência, indo desde a emissão de ruídos até a replicação de pacotes de desautenticação da rede sem fio.

Além disso, outras vulnerabilidades podem ser exploradas nos casos em que as

equipes não se preocupam ou desconsideram uma boa configuração do ponto de acesso. Assim, os ataques de falta de configuração (*Misconfiguration Attacks*), acontecem quando um roteador é configurado usando a configuração padrão, credenciais fracas ou algoritmos de criptografia fracos, podendo o invasor adentrar facilmente na rede e proporcionar instabilidade ou completa ausência de comunicação [Eshete et al. 2011]. Além destes, devido as restrições de processamento comuns aos dispositivos IoT, este cenário é ideal para o ataque do gêmeo do mal [Yang et al. 2012]. Este ataque tem por objetivo comprometer a conexão a partir da configuração de um ponto de acesso falso, com o mesmo SSID. Nesse sentido, o invasor que define configurações semelhantes ao de um AP seria capaz de solicitar detalhes da autenticação do ponto de acesso original e, assim, controlar as funções dos robôs em todo o ambiente.

4. Discussões

As falhas de comunicação que podem ocorrer em eventos como competições de robótica são apenas um exemplo de falhas em redes sem fio que podem ocorrer com qualquer dispositivo IoT com pouca ou nenhuma segurança. A pilha de protocolos TCP/IP praticamente não possui segurança implementada em seus protocolos de baixo nível, deixando a responsabilidade de garantir segurança da comunicação para a camada de aplicação. Com o tempo, os principais protocolos de aplicação não seguros foram substituídos por protocolos seguros, como a troca do TELNET pelo SSH, HTTP por HTTPS, entre outros. No entanto, com o surgimento da IoT, a computação e a comunicação em rede parece muitas vezes ter voltado à condição anterior de ser um ambiente ingênuo em relação à segurança, talvez por estar conectado atrás de um NAT ou em uma LAN.

As competições de robótica são exemplos de ambientes em que as conexões embarcadas geralmente acabam ignorando as questões de segurança para tentar melhorar o desempenho da comunicação. Todas as verificações de segurança usualmente são deixadas de lado pois atrasos de pacotes não são toleráveis neste cenário. Assim, não é comum nesses ambientes verificações sobre o remetente do pacote, ou controle de ordenação dos pacotes recebidos. No entanto, assim como aconteceu com equipes como a UaiSoccer VSS, a preocupação com o desempenho acabou por comprometer o funcionamento do sistema e robôs passaram a atuar de maneira incoerente com o estado atual do ambiente.

Para diferir ataques de causas naturais, a organização do evento poderia monitorar o tráfego da rede para que fosse possível uma auditoria e, assim, possibilitar a detecção de problemas. Já para evitar falhas na comunicação, alterações nas frequências de comunicação poderiam ser realizadas e, assim, reduziriam a interferência de comunicação. Este é o caso da comunicação em 5GHz que, por não ser utilizada por dispositivos Bluetooth e telefones sem fio, por exemplo, é menos suscetível à interferências. Nesse sentido, recorrer diretamente às camadas mais baixas da pilha TCP/IP, como, por exemplo, a camada de enlace, também é um ponto a ser considerado, pois assim seria reduzida a complexidade dos pacotes como também a influência de outras redes na comunicação entre os robôs.

5. Conclusão

O número de dispositivos capazes de se comunicarem através da rede e fornecerem soluções e inovações para as mais diversas aplicações tem crescido nos últimos tempos. No entanto, grande parte das aplicações envolvendo sistemas embarcados e IoT possuem

limitação física, de processamento e memória [Satyadevan et al. 2015], o que pode resultar em um sistema que não leva em consideração as questões de segurança para tentar garantir um melhor desempenho em sua comunicação.

Neste artigo, apresentamos uma situação que faz o uso de sistemas embarcados que, por não possuir preocupações explícitas em relação a segurança, pode estar suscetível a ataques, além de sofrer com causas naturais. Apesar de esta parecer uma situação isolada, tal cenário pode ser expandido para outros cenários, como escritórios e condomínios residenciais, onde a quantidade de dispositivos competindo pelo mesmo canal de transmissão tem aumentado dia após dia.

Este cenário pode ilustrar o tipo de falha de comunicação e sincronização que pode ocorrer em uma situação semelhante, como, por exemplo, a expansão da internet das coisas em todos os ambientes cotidianos e os possíveis ataques e falhas que estes dispositivos podem sofrer. Grandes transtornos podem ocorrer se situação semelhante acontecer com câmeras de segurança e monitoramento de veículos autônomos, por exemplo. Nestas situações, fica evidente a necessidade do uso de protocolos de rede que proporcionem segurança e eficiência em diferentes enlaces, pois a Internet, em princípio, não é segura.

Para trabalhos futuros, pretende-se utilizar ferramentas capazes de analisar por completo o tráfego de informações durante as competições a partir de log de dados. Isso proporcionará análise e identificação das sobrecargas e vulnerabilidades encontradas no cenário aqui apresentado. Além disso, o monitoramentos de tráfego em eventos similares seria adequado para a análise posterior dos dados da comunicação. Estes dados permitiriam também a criação de punições em casos de ataques e criação de regras referentes à rede para que não ocorram problemas.

Agradecimentos

Os autores gostariam de agradecer a Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) sob concessão No.: APQ-03120-17. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

Buttazzo, G. C. (2011). *Hard real-time computing systems: predictable scheduling algorithms and applications*, volume 24. ElsevierSpringer Science & Business Media.

Eshete, B., Villafiorita, A., and Weldemariam, K. (2011). Early detection of security misconfiguration vulnerabilities in web applications. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 169–174. IEEE.

IEEE, L. A. (2009). Ieee very small size rules. *CBR Robótica*.

Satyadevan, S., Kalarickal, B. S., and Jinesh, M. K. (2015). "security, trust and implementation limitations of prominent iot platforms". In *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014*, pages 85–95, Cham. Springer International Publishing.

Yang, C., Song, Y., and Gu, G. (2012). Active user-side evil twin access point detection using statistical techniques. *IEEE Transactions on Information Forensics and Security*, 7(5):1638–1651.

Breve Comparação de Ferramentas para Análise Estática de Código Malicioso

William Akihiro Alves Aisawa[1],
Pedro Henrique Lara Campos[1],
Cesar Augusto Cavalheiro Marcondes[2]
Paulo Matias[1]

[1] Departamento de Computação
Universidade Federal de São Carlos (UFSCar) – São Carlos, SP – Brasil

alveswill@hotmail.com,root@pedrohlc.com,matias@ufscar.br

[2]Divisão de Ciência da Computação
Instituto Tecnológico de Aeronáutica (ITA) – São José dos Campos, SP – Brasil

cmarcondes@ita.br

Abstract. *This article briefly compares IDA Pro and Ghidra, tools for static analysis of software through reverse engineering, focusing on their utility for preventing and counteracting malware, analyzing the code, understanding its operating logic, algorithms, and specifications without risking exposure to the execution of malicious code. Both tools are excellent for static analysis of malware, even Ghidra being a newly released tool.*

Resumo. *Neste artigo, fazemos uma breve comparação entre o IDA Pro e o Ghidra, ferramentas de análise estática por engenharia reversa, focando em sua utilidade para prevenir e contra-atacar malware, analisando os códigos, entendendo sua lógica de funcionamento, os algoritmos e especificações, sem arriscar-se à exposição da execução do código malicioso. Ambas as ferramentas são excelentes para análise estática de malware, mesmo Ghidra sendo uma ferramenta recentemente lançada.*

1. Introdução

Existem grandes indústrias comerciais que giram em torno da criação de *malwares* e *anti-malwares* [Gutmann 2007] e, atualmente, *malwares* formam a parcela mais abundante de diversas categorias de ameças ao ambiente computacional [Zolkipli and Jantan 2010]. Esse fato deve-se ao aprimoramento das técnicas empregadas pelos cibercriminosos, dando origem a diversas categorias de *malware*, que utilizam variados vetores de infecção e apresentam diferentes formas de comportamento errático.

Um dos grandes desafios na análise de *malware* está relacionado à coleta de dados úteis sem arriscar o ambiente computacional usado na análise [Burji et al. 2010]. Precisa-se entender a lógica de funcionamento, algoritmos e especificações sem que o código seja acionado e prejudique o analista. Visando esse tipo de tarefa, existem ferramentas que auxiliam na engenharia reversa, análise e compreensão. Este artigo realiza um breve

estudo comparando duas dessas ferramentas: o **IDA Pro**[1], desenvolvido pela Hex-Rays, e o **Ghidra**[2], recentemente disponibilizado pela *National Security Agency* (NSA).

O artigo está estruturado em três seções. Esta introdução é seguida da Seção 2, que apresenta um resumo de cada uma das ferramentas comparadas e os resultados obtidos na execução de um experimento. Por fim, a Seção 3 apresenta a conclusão do trabalho.

2. Estudo Comparativo e Conceitual das Ferramentas

2.1. Principais Características

As principais características das ferramentas propostas são comparadas na Tabela 1 e introduzidas nos parágrafos seguintes.

	IDA Pro + Hex-Rays	Ghidra
Disassembler	✓	✓
Descompilador	✓	✓
Sistemas Operacionais	Linux, Mac OS X, Microsoft Windows	Linux, Mac OS X, Microsoft Windows
Código Aberto	✗	✓
Grafo de Controle de Fluxo	✓	✓
Extensões	✓	✓
Depurador	✓	✗
Depurador Remoto	✓	✗
Trabalho Colaborativo	✗	✓
Suporte nativo a arquiteturas	60	20

Tabela 1. Comparação entre IDA Pro e Ghidra

O IDA Pro é uma das ferramentas de engenharia reversa mais conhecidas no mercado. Trata-se de um *disassembler* interativo, amplamente utilizado para a reversão de software. Escrito totalmente em C++, é multiplataforma (Linux, Mac OS X e Microsoft Windows) e também possui um grande número de extensões que permitem ampliar suas funcionalidades, como o Hex-Rays Decompiler [Hex-Rays 2019], fornecido pelo seu próprio fabricante.

Ghidra, por sua vez, é um software *open source* de engenharia reversa desenvolvido em Java pelo departamento de pesquisas da NSA. Essa ferramenta também é multiplataforma (oficialmente, suporta Linux, Mac OS X e Microsoft Windows) e ajuda a analisar

[1] `https://www.hex-rays.com/products/ida/index.shtml`
[2] `https://www.nsa.gov/resources/everyone/ghidra`

códigos maliciosos diversos, podendo fornecer aos profissionais uma melhor compreensão das vulnerabilidades potenciais em suas redes e sistemas [National Security Agency 2019].

Disassembler: O IDA conta com suporte a cerca de 60 tipos de arquitetura de microprocessadores, embora sua versão *starter* possua suporte somente a endereçamento de 32-bits. O Ghidra suporta nativamente apenas 20 tipos de arquiteturas. No IDA, pode-se acrescentar suporte a novos conjuntos de instruções por meio de extensões escritas em C ou Python. Já o Ghidra facilita essa tarefa oferecendo uma linguagem de propósito específico denominada Sleigh.

Descompilador: O Ghidra é capaz de descompilar código de qualquer arquitetura para a qual tenha sido adicionado suporte ao *disassembler* por meio da linguagem Sleigh. Já o Hex-Rays Decompiler do IDA só descompila código de algumas arquiteturas explicitamente suportadas pelo fabricante (atualmente, x86, x64, ARM32, ARM64 e PowerPC) e não torna possível que o usuário adicione suporte a uma nova arquitetura.

Extensões: Ambos permitem que extensões sejam desenvolvidas em Python. A implementação de Python utilizada pelo Ghidra é o Jython, que fornece acesso à hierarquia de classes completa do Ghidra, exceto pelo descompilador, que é desenvolvido em código nativo. Apesar disso, pode-se interagir com o descompilador por meio da API.

Depurador: Ao passo que o Ghidra não funciona como depurador, o IDA conta com um depurador versátil, com suporte a múltiplos alvos de depuração e que pode manipular arquivos remotamente por meio de um servidor remoto de depuração. O servidor pode ser executado em uma plataforma diferente da plataforma em que o IDA está sendo executado, tornando possível, por exemplo, depurar o *kernel* Linux de um dispositivo embarcado a partir de um computador com Windows. O depurador do IDA disponibiliza uma API que pode ser manipulada por extensões e *scripts*.

Versionamento e trabalho colaborativo: O Ghidra conta com recursos de versionamento de projeto e trabalho colaborativo, funcionalidades não disponíveis no IDA.

Custo: O Ghidra está disponível para *download* sem custo algum em seu site oficial e seu código fonte está disponível em um repositório no GitHub[3]. Já a versão mais barata do IDA custa em torno de 979 dólares, podendo chegar perto dos 20.000 dólares, uma vez que os descompiladores são vendidos separadamente. A Hex-Rays disponibiliza gratuitamente para uso não-comercial uma versão antiga (atualmente 7.0) do IDA, porém sem descompilador, sem suporte a diversas arquiteturas e formatos de executável e sem suporte a extensões em Python.

Outros recursos: O Ghidra sincroniza o cursor posicionado sobre a saída do descompilador com o cursor na saída do *disassembler*. No entanto, uma funcionalidade similar pode ser obtida no IDA Pro com a extensão HexRaysCodeXplorer[4].

2.2. Breve Comparação Prática

Para comparar a qualidade do código descompilado, uma mesma função de um *malware* foi fornecida a ambas as ferramentas (Figura 1). O malware utilizado possui o resumo (*hash*) `551b48e425dcf4337ee023ad65a871123d172e43fabbc965252f5a2e69d0bd4a`.

[3] `https://github.com/NationalSecurityAgency/ghidra`
[4] `https://github.com/REhints/HexRaysCodeXplorer`

```
/*Código descompilado pelo IDA Pro*/
_BOOL4 __cdecl
CUtility::IpIsLan(CUtility *this)
{
  unsigned int v1; //eax
  unsigned int v2; //eax
  unsigned int v3; //eax
  unsigned int v4; //eax
  unsigned int v5; //eax
  unsigned int v6; //eax
  unsigned int v7; //eax
  unsigned int v8; //eax
  //[esp-8h][ebp-10h]
  unsigned int v10;
  //[esp-8h][ebp-10h]
  unsigned int v11;
  //[esp-8h][ebp-10h]
  unsigned int v12;
  //[esp-8h][ebp-10h]
  unsigned int v13;
  //[esp-8h][ebp-10h]
  unsigned int v14;
  bool v16;//[esp+4h][ebp-4h]
  bool v17;//[esp+5h][ebp-3h]
  bool v18;//[esp+6h][ebp-2h]
  bool v19;//[esp+7h][ebp-1h]

  if ( !this )
    return 1;
  v1 = inet_addr("10.0.0.0");
  v16 =
  CUtility::IpCompare(this, v1, v10) < 0
  && (v2 = inet_addr("10.255.255.255"),
  CUtility::IpCompare(this, v2, v11)>0)
  if ( v16 )
    return 1;
}
```

```
/*Código descompilado pelo Ghidra*/
uint IpIsLan(uint param_1)
{
  byte bVar1;
  bool bVar2;
  in_addr_t iVar3;
  int iVar4;
  uint local_c;

  if (param_1 == 0) {
    return 1;
  }
  iVar3 = inet_addr("10.0.0.0");
  iVar4 = IpCompare(param_1,iVar3);
  if (iVar4 < 0) {
    iVar3 = inet_addr("10.255.255.255");
    iVar4 = IpCompare(param_1,iVar3);
    if (iVar4 < 1) goto LAB_0808ec2a;
    bVar2 = true;
  }
  else {
LAB_0808ec2a:
    bVar2 = false;
  }
  if (bVar2) {
    return 1;
  }
}
```

Figura 1. Códigos descompilados pelo IDA Pro com Hex-Rays Decompiler e pelo Ghidra

O Ghidra reconheceu corretamente o tipo da variável `iVar3`, ao passo que no Hex-Rays seria necessário corrigir a assinatura do método `IpCompare` para obter um código igualmente legível. Esse comportamento ajuda na análise de *malware* pois desofusca estruturas como a `in_addr_t`, que armazena endereços resolvidos em conexões de rede, facilitando assim a localização e a compreensão de como as conexões são formadas e utilizadas.

O Hex-Rays conseguiu expressar o fluxo de execução inteiramente com construções de programação estruturada, ao passo que o Ghidra gerou código que faz uso de `gotos`. O primeiro é mais eficiente para uso em análises, já que a instrução `goto` é muito evitada e quando aparece demonstra menor fidelidade ao código original.

As ferramentas constroem condições de formas diferentes: o Hex-Rays tenta produzir o menor código possível. No exemplo, a diferença por zero foi substituída por uma simples negação, alguns blocos de condição com atribuições de valores por condições lógicas e, quando o efeito da condição foi de apenas uma linha, os colchetes foram omitidos. Neste caso, considerações a respeito da melhor legibilidade são mais subjetivas, variando de acordo com a pessoa que fará a analise e com o código a ser analisado.

O Ghidra permite que mais de um arquivo seja analisado no mesmo projeto, assim bibliotecas dinâmicas e até dependências puderam ser adicionadas para compreender

melhor as chamadas à funções externas e sua participação final no objetivo do código malicioso.

A visualização do código lado-a-lado com o *assembly* do Ghidra permite uma consulta prática em casos que o descompilador não produziu um resultado claro, útil em casos de ofuscação que tenha como alvo esse tipo de análise.

O Hex-Rays possui um grafo de chamadas entre as funções num escopo do aplicativo inteiro enquanto no Ghidra o grafo fica restrito à função selecionada. Esses grafos auxiliam de forma diferentes na compreensão do funcionamento de um *malware*.

Ferramentas básicas como busca por chamadas de funções, símbolos e *strings*, manipulação dos tipos de dados, comentários, rótulos, manuais e informações de ajuda estão presentes em ambas as ferramentas.

Verificou-se que o tempo para abertura e análise do *malware* no IDA Pro foi menor que no Ghidra. Para análise manual por parte de um analista, a diferença pode ser pouco impactante, uma vez que essa espera só é necessária quando um binário é adicionado ao projeto. No entanto, a diferença no tempo de análise pode ser significativa para pesquisas que envolvam a automação de análise de uma grande amostragem de *malwares*.

3. Conclusões

Apesar de recentemente lançado, o Ghidra apresentou-se como ferramenta competitiva para análise estática de binários com relação ao IDA Pro. Em termos de extensibilidade, o Ghidra já apresenta vantagens, por ser de código aberto e por possibilitar acrescentar suporte a novas arquiteturas inclusive em seu descompilador. O presente estudo limitou-se a apresentar um resumo das características divulgadas pelos próprios fabricantes das ferramentas, além de algumas evidências anedóticas de diferenças entre as funcionalidades de cada uma. Espera-se que estudos futuros abordem de maneira mais detalhada a qualidade da saída dos descompiladores, colaborando para a evolução das ferramentas e para seu potencial de acelerar o trabalho do analista de *malware*.

Referências

Burji, S., Liszka, K. J., and Chan, C.-C. (2010). Malware analysis using reverse engineering and data mining tools. In *2010 International Conference on System Science and Engineering*, pages 619–624. IEEE.

Gutmann, P. (2007). The commercial malware industry. In *DEFCON Conference*.

Hex-Rays (2019). Hex-rays. `https://www.hex-rays.com/files/hexrays_info.pdf`.

National Security Agency (2019). Ghidra. `https://www.nsa.gov/resources/everyone/ghidra`.

Saxe, J. and Sanders, H. (2018). *Malware Data Science: Attack Detection and Attribution*. No Starch Press.

Zolkipli, M. F. and Jantan, A. (2010). Malware behavior analysis: Learning and understanding current malware threats. In *2010 Second International Conference on Network Applications, Protocols and Services*, pages 218–221. IEEE.

II Workshop de Segurança Cibernética em Dispositivos Conectados
SBRC 2019
Sessão Técnica 3

Modelo de maturidade de segurança cibernética para os órgãos da Administração Pública Federal

Antonio João Gonçalves de Azambuja[1], João Souza Neto[2]

[1] Mestre em Gestão do Conhecimento e Tecnologia da Informação pela Universidade Católica de Brasília (UCB). Doutorando da Universidade Federal do Rio Grande do Sul (UFRGS). Chefe do Serviço de Segurança da Informação e Comunicações da Advocacia-Geral da União (AGU).
[2] Doutor em Engenharia Elétrica pela Universidade de Brasília (UnB), Professor do Mestrado de Governança, Tecnologia e Inovação da Universidade Católica de Brasília (UCB).

{ajaazambuja@gmail.com, sznetoj@gmail.com}

Abstract. *This paper presents a Cybersecurity maturity model for the agencies of the Brazilian Federal Public Administration. Qualitative research was conducted to analyze Cybersecurity maturity models found in the literature, which served as ground to develop the proposed model. To analyze, understand and construe the qualitative material, we used content analysis and an online questionnaire as technical procedures. The content analysis was divided into pre-analysis, material exploration and handling of results which allowed setting the domains of the proposed model. The model was applied through an online questionnaire to some agencies of the Brazilian Federal Public Administration. The results evidenced that, in general, the agencies surveyed have low maturity in Cybersecurity.*

Resumo. *O artigo apresenta um modelo de maturidade de Segurança Cibernética (SegCiber) para os órgãos da Administração Pública Federal (APF). Foi realizada uma pesquisa qualitativa para analisar os modelos de maturidade de SegCiber encontrados na literatura, que foram a base para o desenvolvimento do modelo proposto. Para analisar, compreender e interpretar o material qualitativo, os procedimentos técnicos utilizados foram a análise de conteúdo e um questionário online. A análise de conteúdo foi dividida na fase de pré-análise, exploração do material e tratamento dos resultados, que permitiu a definição dos domínios do modelo proposto. Os resultados da aplicação do modelo, por meio do questionário online, demonstram que, no geral, há baixa maturidade dos órgãos pesquisados.*

1. Introdução

A informação tem se mostrado, nos dias atuais, um ativo de valor para as organizações, talvez o mais precioso dada a sua importância para os negócios, portanto, deve ser protegida.

Os bens essenciais para o funcionamento de uma sociedade, como as redes de computadores, sistemas de informação, de transporte, financeiros, de saúde, entre outros, estão cada vez mais dependentes da Tecnologia da Informação (TI) (Rahman *et al*., 2011, Xiao-Juan & Li-Zhen, 2010).

A Estratégia de Segurança da Informação e Comunicações e de Segurança

Cibernética da Administração Pública Federal (BRASIL, 2015) define a Segurança Cibernética (SegCiber) como a arte de assegurar a existência da sociedade da informação de uma nação, garantindo e protegendo, no espaço cibernético, seus ativos de informação e suas infraestruturas críticas.

Para a norma ABNT NBR ISO/IEC 27032:2015, as práticas básicas de Segurança da Informação (SI) voltadas para as partes interessadas no espaço cibernético fornecem as diretrizes para melhorar o estado de SegCiber, determinando os aspectos comuns dessa atividade e suas ramificações em outros domínios de segurança, tais como: as redes computadores e a proteção de infraestruturas críticas de informação (ABNT, 2015).

As ações de Segurança da Informação (SI) têm como objetivo a proteção das informações de vários tipos de ameaças para garantir a continuidade, minimizar os riscos e maximizar o retorno sobre os investimentos e as oportunidades de negócio (Manoel, 2014).

O espaço cibernético constitui novo e promissor cenário para a prática de toda a sorte de atos ilícitos, desafiando conceitos tradicionais, entre eles, o de fronteiras geopolíticas e organizacionais, constituindo novo território, por vezes desconhecido, a ser desbravado pelos bandeirantes do século XXI (Machado, 2010).

A avaliação da SegCiber nas organizações pode ser realizada por meio de um modelo de maturidade, que fornece um ponto de referência para conhecer o nível de suas práticas, processos e métodos para, então, definir metas e prioridades de melhoria.

Do ponto de vista da SegCiber, deve-se adotar ações que assegurem a disponibilidade, integridade, confidencialidade e autenticidade das informações de interesse do Estado brasileiro (Mandarino Júnior, 2010), os quais são princípios básicos da SI.

Entre as metas descritas na atual Estratégia de Segurança da Informação e Comunicações e de Segurança Cibernética da APF, publicada pelo Gabinete de Segurança Institucional da Presidência da República (GSI/PR), destaca-se como mecanismo de acompanhamento e avaliação de SegCiber nos órgãos da APF, conhecer e implementar o indicador anual do nível de maturidade na área.

Diante do exposto, este artigo busca apresentar um modelo de maturidade de SegCiber para ajudar as organizações públicas a avaliarem o seu estado atual da SegCiber e assegurar a continuidade do negócio. Para tal, apresenta um referencial teórico, metodologia utilizada na pesquisa e os resultados da aplicação do modelo com a participação de 35 organizações.

2. Referencial Teórico

2.1. Estratégia de Segurança da Informação e Comunicações e SegCiber da APF

A Estratégia é um instrumento de apoio ao planejamento estratégico governamental que reúne um conjunto de objetivos estratégicos e metas para o período de 2015 a 2018, elaborada pelo Gabinete de Segurança Institucional da Presidência da República (GSI/PR).

Considerando que são atividades estratégicas para o Estado, essa Estratégia tem como objetivo a articulação e coordenação de esforços dos diversos atores envolvidos, de forma a atingir o aprimoramento das ações de segurança e resiliência das infraestruturas

críticas, dos serviços de Estado e a mitigação dos riscos aos quais encontram-se expostas as organizações e a sociedade (BRASIL, 2015).

A Estratégia ressalta que não obstante os esforços do Governo em fortalecer as ações de SIC e de SegCiber, o que inclui o arcabouço de normas complementares publicadas pelo GSI desde 2008, no geral, os níveis de maturidade dos órgãos da APF ainda se encontram em patamar aquém do desejado (BRASIL, 2015).

2.2. Norma ISO/IEC 27032:2012

A Norma ISO/IEC 27032:2012 estabelece diretrizes para melhorar o estado de SegCiber de uma organização, traçando os aspectos típicos desta atividade e suas ramificações em outros domínios de segurança.

Aborda a SegCiber como a preservação da confidencialidade, integridade e disponibilidade da informação no ciberespaço. Define como ciberespaço um ambiente complexo resultante da interação de pessoas, software e serviços da Internet por meio de dispositivos tecnológicos e redes conectadas, que não existe em nenhum formato físico (ABNT, 2015).

2.3. Modelos de maturidade de SegCiber

2.3.1. *Cybersecurity Capability Maturity Model* (C2M2)

O *Cybersecurity Capability Maturity Model* (U.S. Department of Energy, 2014) pode ajudar as organizações de todos os setores, tipos e tamanhos a avaliar e fazer melhorias em seus programas de SegCiber. O foco está na implementação de práticas e gestão de SegCiber associadas aos ativos de TI, operações de tecnologia e o seu ambiente de operação.

As organizações podem usar esse documento para os seguintes fins: **i)** fortalecer as capacidades de SegCiber; **ii)** permitir a avaliação de forma eficaz e consistente do estado atual da SegCiber; **iii)** compartilhar conhecimento, melhores práticas e referências de SegCiber; e **iv)** permitir a priorização das ações e investimentos para melhorar a SegCiber.

O modelo apresenta uma metodologia de autoavaliação nas organizações, visando a identificação dos seus níveis de maturidade e as melhorias a serem realizadas no programa de SegCiber. A autoavaliação fornece informações aos seguintes atores: gestores responsáveis pela tomada de decisão; gestores responsáveis pela gestão de recursos e operações organizacionais; gestores responsáveis pela aplicação da autoavaliação; e facilitadores da aplicação de autoavaliação.

O C2M2 usa uma escala de quatro níveis, que permite à organização definir o seu estado atual de SegCiber, determinar o futuro desejado e identificar os recursos necessários para alcançar esse estado futuro.

2.3.2. *NIST Cybersecurity Framework*

O modelo foi desenvolvido em resposta a uma ordem do Presidente dos EUA, Barack Obama, em fevereiro de 2013, para reforçar a resiliência da infraestrutura crítica daquele país e manter um ambiente cibernético que encoraje a eficiência, inovação e prosperidade econômica (The President, 2013).

O *NIST Cybersecurity Framework* (2014) permite que as organizações apliquem os princípios e as melhores práticas de gestão de risco para aprimorar a SegCiber e a resiliência das infraestruturas críticas. Tem as seguintes funções e objetivos: **i)** identificar:

desenvolver a compreensão organizacional para gerenciar o risco de sistemas, ativos, dados e recursos; **ii)** proteger: desenvolver e implementar as salvaguardas adequadas para assegurar os serviços de infraestrutura crítica; **iii)** detectar: desenvolver e implementar as atividades apropriadas para identificar a ocorrência de eventos de SegCiber; **iv)** responder: desenvolver e implementar as atividades apropriadas para tomar medidas relativas a eventos de SegCiber detectados; e **v)** recuperar: desenvolver e implementar as atividades apropriadas para manter planos de resiliência e restaurar quaisquer capacidades ou serviços que foram prejudicados devido a eventos de SegCiber.

Os níveis utilizados no modelo são: **i)** parcial; **ii)** informado; **iii)** repetido; e **iv)** adaptado. Esses níveis descrevem um grau crescente de rigor e sofisticação nas práticas de risco de SegCiber e a sua integração com o risco global da organização e as suas necessidades para o negócio.

2.3.3. *The Community Cyber Security Maturity Model*

O *The Community Cyber Security Maturity Model* (CCSMM), proposto por White (2007), fornece uma estrutura que as comunidades e os estados podem usar para determinar seu nível de preparação para criar um plano para melhorar sua postura de SegCiber.

O CCSMM reconhece a necessidade de organizações terem métricas tecnológicas para desenvolver um programa de SegCiber, como também exercitar testes de capacidade de segurança, implementar atividades de treinamento e compartilhamento das informações relacionadas com a SegCiber.

3. Metodologia

Esta pesquisa classifica-se como Pesquisa Aplicada, quanto à sua natureza. Este tipo de pesquisa objetiva gerar conhecimento para aplicação prática, dirigida à solução de problemas específicos.

Com relação à forma de abordagem do problema, foi realizada uma pesquisa qualitativa para analisar, compreender e interpretar os modelos de maturidade de SegCiber encontrados na revisão da literatura, que foram a base para o desenvolvimento do modelo de maturidade de SegCiber para APF, ora proposto.

Do ponto de vista dos objetivos, a pesquisa é exploratória, já que foi realizada uma avaliação dos modelos de maturidade de SegCiber disponíveis na literatura. Para analisar, compreender e interpretar o material qualitativo, os procedimentos técnicos utilizados foram a análise de conteúdo e um questionário *online*.

O questionário, como instrumento de pesquisa, permite a obtenção de dados ou informações sobre as características ou as opiniões de determinado grupo de pessoas, indicado como representante de uma população-alvo (Fonseca, 2002).

A análise de conteúdo, por sua vez, representa um conjunto de técnicas de análise de comunicações que visam a obter, por procedimentos sistemáticos e objetivos de descrição do conteúdo das mensagens, indicadores (quantitativos ou não) que permitam a inferência de conhecimentos relativos às condições de produção ou recepção (variáveis inferidas) destas mensagens (Bardin, 2011).

A análise de conteúdo, conforme Bardin (2011), prevê três fases: **i)** pré-análise,

com as seguintes sub-fases: leitura sobre o tema; seleção do material; representatividade do material; homogeneidade e pertinência do material; **ii)** exploração do material; e **iii)** tratamento dos resultados, inferência e interpretação.

3.1 Descrição da pesquisa

Considerando o método de Bardin (2011), a pesquisa foi dividida em fases. Na primeira, pré-análise, foi realizada a revisão da literatura e a seleção do material utilizado como base de apoio teórico para a pesquisa. Na segunda, foram organizados os dados dos modelos em domínios, identificando as categorias e as unidades de registro. A terceira consistiu no tratamento dos resultados que, para Bardin (2011), compreende a inferência e a interpretação.

3.2 Análise de conteúdo

Este estudo analisou os seguintes modelos *Cybersecurity Capability Maturity Model*, *NIST Cybersecurity Framework* e *The Community Cyber Security Maturity Model*.

3.2.1 Pré-análise

As atividades realizadas foram:

- Revisão da literatura;
- Organização do material; e
- Relacionamentos dos pontos relevantes e comuns entre os modelos.

3.2.2 Tratamento dos resultados

A definição dos temas permitiu o agrupamento em nove domínios, para o modelo proposto, conforme similaridade identificada entre os temas na análise de conteúdo.

Os domínios do modelo proposto são: gestão de riscos; gestão de ativos; gestão de acesso; gestão de ameaças e vulnerabilidades; gestão de continuidade; compartilhamento de informações; capacitação, conscientização e cultura; infraestrutura tecnológica; e governança de SegCiber.

4. Apresentação dos resultados

Os resultados obtidos com a pesquisa possibilitaram a elaboração do Modelo de Maturidade de SegCiber para os órgãos da APF, com nove domínios, 33 objetivos, 56 práticas para o Nível 1 (N-1), 111 práticas para o Nível 2(N-2) e 117 práticas para o Nível 3 (N-3).

4.1 Níveis de maturidade para o modelo proposto

Os níveis são utilizados para medir a competência organizacional ou maturidade de um conjunto reconhecido das melhores práticas. No quadro 1 são apresentados os domínios do modelo proposto e o seu relacionamento com domínios dos modelos estudados. O modelo do C2M2 contribuiu com dez domínios, o modelo do NIST com três e o modelo elaborado por White (2007) com seis.

Quadro 1 - Domínio do modelo proposto vs. modelos analisados

Domínios do modelo proposto	Relacionamento dos domínios dos modelos com os domínios do modelo proposto		
	C2M2	NIST Cybersecurity Framework	The Community Cyber Security Maturity Model
1. Gestão de risco	·	·	
2. Gestão de ativos	·		
3. Gestão de acesso	·		
4. Gestão de ameaças e vulnerabilidades	·		·
5. Gestão de continuidade	·		
6. Compartilhamento de informações	·	·	·
7. Capacitação, conscientização e cultura	·	·	·
8. Infraestrutura tecnológica	·		·
9. Governança de SegCiber	·		
Total de domínios dos modelos	10	3	6
Percentual dos relacionamentos dos domínios com os modelos	47,36% (9)	15,78% (3)	21,05% (4)

Fonte: Os autores

No quadro 1, é possível identificar que o maior percentual de alinhamento dos domínios do modelo proposto ocorre com os domínios do C2M2, com um percentual de 47,36 %, contra 15,78% para o modelo do NIST e 21,05% para o modelo CCSMM.

A justificativa para selecionar os domínios do modelo C2M2 foi o maior percentual de alinhamento entre os domínios do modelo proposto e os domínios do C2M2, conforme apresentado no quadro 1.

4.2 Domínios do modelo de maturidade proposto

Cada um dos nove domínios do modelo proposto contém um conjunto estruturado de objetivos e práticas de SegCiber relacionados com os níveis de maturidade estabelecidos.

Os objetivos dos domínios compreendem um conjunto de práticas, que são ordenadas por nível de maturidade. Um conjunto de práticas representa as atividades que uma organização pode realizar para implementar e desenvolver a capacidade de maturidade em um domínio.

O modelo proposto fornece uma orientação para ajudar as organizações medir e melhorar as suas capacidades de SegCiber com base nos padrões existentes.

4.2.1 Segue uma breve descrição dos domínios do modelo:

- Gestão de riscos: a gestão de riscos em SegCiber fornece orientação para analisar e priorizar o risco e define a tolerância ao risco;
- Gestão de ativos: realizar a gestão adequada de todos os ativos, evita a perda de informações, otimiza as atividades do negócio, assegura a confiabilidade,

integridade e disponibilidade da informação;

- Gestão de acesso: a gestão de acesso propõe a criação e gerenciamento de identidades para acesso lógico e físico aos ativos de informação. Práticas inadequadas de gerenciamento de acesso podem levar ao uso, divulgação, modificação e destruição não autorizada da informação;
- Gestão de ameaças e vulnerabilidades: a gestão de ameaças e vulnerabilidades visa estabelecer e manter planos, procedimentos e tecnologias para detectar, identificar, analisar, gerenciar e responder a ameaças e vulnerabilidades de SegCiber;
- Gestão de continuidade: a gestão de continuidade procura estabelecer e manter planos, procedimentos e tecnologias para detectar, analisar e responder a eventos de SegCiber para sustentar as operações da organização;
- Compartilhamento de informações: visa estabelecer e manter relações com entidades internas e externas para coletar e compartilhar informações sobre SegCiber;
- Capacitação, conscientização e cultura: procura criar uma cultura de SegCiber na organização e assegurar a adequação das competências atualizadas para a força de trabalho;
- Infraestrutura tecnológica: requer uma infraestrutura tecnológica com mecanismos para identificar, tratar e responder as ameaças e vulnerabilidades de forma integrada; e
- Governança de SegCiber: visa estabelecer e manter um programa corporativo de SegCiber que forneça governança, planejamento estratégico e patrocínio para as atividades de SegCiber.

4.3 Estrutura do modelo proposto

As práticas para cada um dos domínios estão agrupadas por objetivos que apoiam a estrutura do modelo. Na tabela 1 estão apresentados os nove domínios, trinta e três objetivos e o número de práticas para os seus respectivos níveis de maturidade.

Tabela 1 - Domínios vs Objetivos vs Práticas vs Níveis de Maturidade

Domínios / Objetivos	Número de práticas			
	Nível 0	Nível 1	Nível 2	Nível 3
Gestão de riscos				
Estabelecer a estratégia de gestão de risco	Não tem práticas	2	2	3
Gerenciar o risco de SegCiber		2	5	3
Realizar atividades de gestão		1	4	5
Gestão de ativos				
Gerenciar inventário de ativos		1	2	2
Gerenciar a configuração de ativos		1	1	2
Gerenciar as mudanças nos ativos		2	2	2
Realizar atividades de gestão		1	4	5

Gestão de acesso				
Estabelecer e manter identidades	Não tem práticas	3	3	1
Controlar acesso		3	3	3
Realizar atividades de gestão		1	4	5
Gestão de ameaças e vulnerabilidades				
Identificar e responder a ameaças		3	3	3
Reduzir as vulnerabilidades de SegCiber		3	5	6
Realizar atividades de gestão		1	4	5
Gestão de continuidade				
Detectar eventos de SegCiber		2	2	3
Escalar eventos e classificar incidentes de SegCiber		3	4	2
Responder a incidentes e eventos escalados de SegCiber		3	4	5
Elaborar e manter plano de continuidade		3	4	4
Realizar atividades de gestão		1	4	5
Compartilhamento de informações				
Compartilhar informações sobre SegCiber		2	5	4
Realizar atividades de gestão		1	4	6
Capacitação, conscientização e cultura				
Atribuir responsabilidades de SegCiber		2	2	3
Controlar o ciclo de vida da força de trabalho		2	2	4
Desenvolver a força de trabalho de SegCiber		1	3	5
Aumentar a conscientização em SegCiber		1	2	2
Realizar atividades de gestão		1	4	5
Infraestrutura tecnológica				
Realizar e monitorar as atividades operacionais		2	4	5
Estabelecer e manter um painel operacional padrão		1	3	3
Realizar atividades de gestão		1	4	5
Governança de SegCiber				
Estabelecer a estratégia do programa de SegCiber		1	5	1
Patrocinar o programa de SegCiber		2	6	3
Estabelecer e manter a arquitetura de SegCiber		1	2	1
Realizar o desenvolvimento de software seguro		1	1	1
Realizar atividades de gestão		1	4	5
Totais: 9 domínios, 33 objetivos	0	56	111	117

Fonte: Os autores

Para realizar a avaliação, foi construído um questionário *online* com os domínios, objetivos e práticas que devem ser implementadas pela organização para identificação do seu nível de maturidade. A avaliação permite a identificação das lacunas na maturidade da organização.

Para aplicação do modelo, a organização deve selecionar um avaliador que esteja familiarizado com a estrutura do modelo e tenha, também, conhecimento para ajudar a organização a entender os seus objetivos e realizar um planejamento para atingir níveis mais elevados de maturidade.

4.4 Resultados da aplicação do modelo

Para aplicação do modelo, foram selecionadas 35 (trinta e cinco) organizações da APF, que relataram a necessidade de assegurar a confidencialidade, integridade e disponibilidade das suas informações, garantindo a proteção dos seus ativos de informação e das suas infraestruturas críticas para continuidade do negócio.

A tabela 2 apresenta o percentual das organizações por nível para cada um dos objetivos dos domínios do modelo proposto.

Tabela 2 - Percentuais dos objetivos

Domínios	Percentuais			
	Nível 0	Nível 1	Nível 2	Nível 3
Gestão de Risco)	34,38	26,56	18,75	20,31
Estabelecer a estratégia de gestão de risco	43,75	18,75	12,50	25,00
Gerenciar o risco de SegCiber	31,25	37,50	18,75	12,50
Gerenciar o risco de SegCiber	31,25	37,50	18,75	12,50
Realizar atividades de gestão	31,25	12,50	25,00	31,25
Gestão de Ativos	46,88	28,13	7,81	17,19
Gerenciar inventário de ativos	31,25	43,75	6,25	18,75
Gerenciar a configuração de ativos	56,25	18,75	6,25	18,75
Gerenciar as mudanças nos ativos	56,25	18,75	12,50	12,50
Realizar atividades de gestão	43,75	31,25	6,25	18,75
Gestão de Acesso	47,92	27,08	8,33	16,67
Estabelecer e manter identidades	31,25	43,75	6,25	18,75
Controlar acesso	56,25	18,75	6,25	18,75
Realizar atividades de gestão	56,25	18,75	12,50	12,50
Gestão de Ameaças e Vulnerabilidades	43,75	25,00	16,67	14,58
Identificar e responder a ameaças	43,75	31,25	6,25	18,75
Reduzir as vulnerabilidades de SegCiber	43,75	12,50	31,25	12,50
Realizar atividades de gestão	43,75	31,25	12,50	12,50
Gestão de Continuidade	47,50	26,25	10,00	16,25
Detectar eventos de SegCiber	37,50	31,25	6,25	25,00
Escalar eventos e classificar incidentes de SegCiber	68,75	6,25	12,50	12,50
Responder a incidentes e eventos escalados de SegCiber	37,50	37,50	12,50	12,50
Domínios	Percentuais			
	Nível 0	Nível 1	Nível 2	Nível 3
Elaborar e manter plano de continuidade	43,75	31,25	12,50	12,50

Realizar atividades de gestão	50,00	25,00	6,25	18,75
Compartilhamento de Informações	40,63	28,13	6,25	25,00
Compartilhar informações sobre SegCiber	37,50	25,00	12,50	25,00
Realizar atividades de gestão	43,75	31,25	0,00	25,00
Capacitação, Conscientização e Cultura	42,50	26,25	15,00	16,25
Atribuir responsabilidades de SegCiber	37,50	18,75	25,00	18,75
Controlar o ciclo de vida da força de trabalho	31,25	25,00	31,25	12,50
Desenvolver a força de trabalho de SegCiber	56,25	25,00	6,25	12,50
Aumentar a conscientização em SegCiber	37,50	31,25	12,50	18,75
Realizar atividades de gestão	50,00	31,25	0,00	18,75
Infraestrutura Tecnológica	47,92	12,50	18,75	20,83
Realizar e monitorar as atividades operacionais	31,25	12,50	25,00	31,25
Estabelecer e manter um painel operacional padrão	68,75	0,00	12,50	18,75
Realizar atividades de gestão	43,75	25,00	18,75	12,50
Governança de SegCiber	62,50	8,75	11,25	17,50
Estabelecer a estratégia do programa de SegCiber	68,75	6,25	12,50	12,50
Patrocinar o programa de SegCiber	68,75	0,00	12,50	18,75
Estabelecer e manter a arquitetura de SegCiber	62,50	6,25	12,50	18,75
Realizar o desenvolvimento de software seguro	68,75	6,25	0,00	25,00
Realizar atividades de gestão	43,75	25,00	18,75	12,50

Fonte: Os autores

Para domínio gestão de riscos, 34,38 % das organizações estão no N-0, ou seja, não realizam práticas de gestão de riscos. O elevado número de organizações no N-0 demonstra a falta de uma estratégia de gestão de riscos de SegCiber.

No domínio gestão de ativos, o percentual de organizações no N-0 foi de 46,88%, maior que no domínio gestão de riscos. Com a falta de maturidade nesse domínio, a organização não consegue assegurar a confiabilidade, integridade e disponibilidade da informação, uma vez que a realização de uma gestão adequada dos ativos evita a perda de informações e otimiza as atividades do negócio.

Os resultados do domínio gestão de acesso apresentam o menor percentual de organizações no N-0 em comparação com os demais. O fato da maioria das organizações estar no N-1, N-2 e N-3 demonstra que existe, por parte das organizações, controles de acesso lógico relacionados com os sistemas de informação e controles de acesso físico às instalações.

No domínio gestão de ameaças e vulnerabilidades a maior parte das organizações pesquisadas está no N-0 para esse domínio. Essas organizações não possuem práticas para este domínio, o que torna difícil e oneroso detectar, identificar, analisar, gerenciar e responder às ameaças e vulnerabilidades de SegCiber.

Os níveis de maturidade das organizações para o domínio gestão de continuidade apresentaram os seguintes percentuais: 47,50 % (N-0), 26,25 % (N-1), 10 % para o (N-2) e 16,25 % (N-3). A baixa maturidade para esse domínio demonstra que as organizações não possuem planos, procedimentos e tecnologias para sustentar as operações das organizações como resposta a eventos de SegCiber.

O maior número das organizações está no N-0 para o domínio compartilhamento de informações. O compartilhamento de informações é um instrumento para aumentar o conhecimento para enfrentar as ameaças e vulnerabilidades (White, 2007).

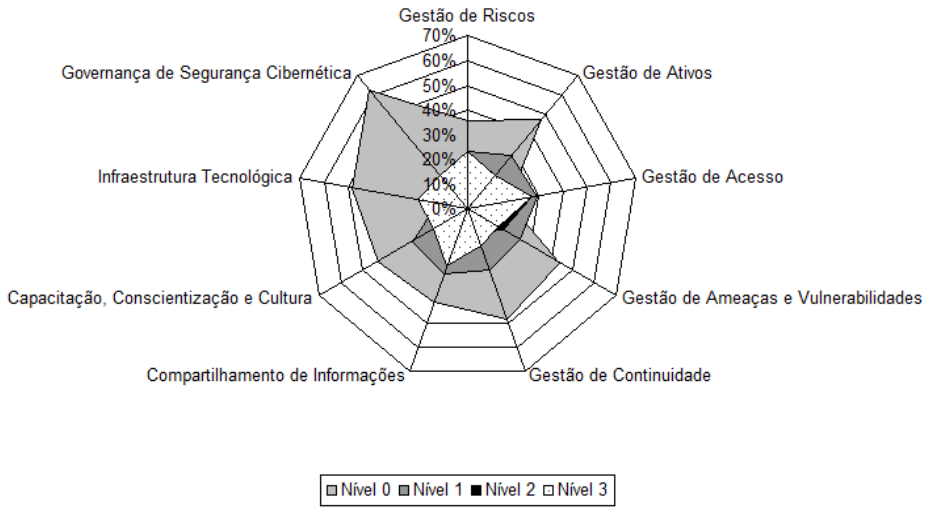
Para o para o domínio capacitação, conscientização e cultura existe um elevado percentual das organizações no N-0. Tal resultado demonstra a falta de práticas para a criação de uma cultura de SegCiber nas organizações. O treinamento e conscientização da força de trabalho são abordagens tão importantes para a SegCiber quanto as questões tecnológicas.

O domínio infraestrutura tecnológica tem um percentual de 47,92 % das organizações no N-0, ou seja, não têm práticas que definem os requisitos de monitoramento e análise dos eventos de SegCiber.

Os percentuais dos níveis de maturidade para o domínio governança de SegCiber demonstram que a maioria das organizações está no N-0 entre todos os domínios. Para o objetivo "patrocinar o programa de SegCiber", 68,75% das organizações estão no N-0, o que significa que as organizações não têm apoio da alta administração para o desenvolvimento e manutenção de políticas de SegCiber.

O Gráfico 1 apresenta o percentual das organizações por nível em cada domínio do modelo.

Gráfico 1 - Percentual das organizações por nível



Fonte: Os autores

5. Conclusão

Este estudo propôs o desenvolvimento de um modelo de maturidade de SegCiber para os órgãos da APF, considerando as características dos modelos apresentados no referencial teórico e as características da Estratégia de SIC e de SegCiber da APF.

Assim sendo, foi realizada a análise de conteúdo com a fase de pré-análise, que foi a revisão da literatura, a fase de exploração do material, na qual foram organizados os domínios dos modelos selecionados e a fase de tratamento dos resultados para a interpretação dos temas e domínios comuns entre os modelos. A definição dos temas possibilitou o agrupamento de nove domínios para o modelo proposto.

O modelo foi submetido aos órgãos participantes por meio de questionário *online*, para uma avaliação da maturidade de SegCiber das organizações, atividade prevista na meta XXI e XXII da Estratégia de Segurança da Informação e Comunicações e de Segurança Cibernética da APF.

Com a aplicação do modelo foi possível identificar, entre as organizações participantes que, no geral, há baixa maturidade dos órgãos pesquisados em SegCiber. Os resultados demandam ações de melhoria por parte das organizações que possuem ativos de informação que devem ser protegidos de ameaças cibernética.

O maior número das organizações está no Nível 0 (N-0) no domínio governança de SegCiber, o que reforça a necessidade de implementar a governança de SegCiber nos órgãos da APF, em consonância com o objetivo IV da Estratégia de SegCiber, que propõe estabelecer um modelo de Governança de Segurança da Informação e Comunicações e Segurança Cibernética na APF.

As organizações da APF devem planejar ações que fortaleçam a defesa nacional para mitigar vulnerabilidades que podem ser exploradas com os avanços tecnológicos no espaço cibernético e o crescimento das interações entre os usuários e a tecnologia, visando a continuidade do negócio na era da sociedade da informação.

Referências

ADLER, Richard M. A Dynamic Capability Maturity Model for Improving Cyber Security. Technologies for Homeland Security (HST), IEEE International Conference on. DecisionPath, Inc. Winchester, MA USA. 2013. Disponível em: <http://ieeexplore.ieee.org/document/6699005/?reload=true>. Acesso em: 10 de janeiro de 2017.

ABNT - NBR ISO/IEC 27032:2012: Tecnologia da Informação - Técnicas de segurança - Diretrizes para segurança cibernética. Rio de Janeiro: ABNT, 2015.

Bardin, l. Análise de conteúdo. São Paulo: Edições 70, 2011.

Brasil. Desafios estratégicos para segurança e defesa cibernética. Secretaria de Assuntos Estratégicos da Presidência da República. Organizadores: Otávio Santana Rêgo Barros, Ulisses de Mesquita Gomes, Whitney Lacerda de Freitas. Brasília, 2011. Disponível em: <http://www.biblioteca.presidencia.gov.br/presidencia/dilma-vana-rousseff/publicacoes/orgao-essenciais/secretaria-de-assuntos-estrategicos/deafios-estrategicos-para-a-seguranca-e-defesa-cibernetica/view>. Acesso em: 05 de abril de 2017.

Brasil. Presidência da República. Gabinete de Segurança Institucional. Estratégia de Segurança da Informação e Comunicações e de Segurança Cibernética da Administração Pública Federal 2015-2018: versão 1.0 / Gabinete de Segurança Institucional, Secretaria-Executiva, Departamento de Segurança da Informação e Comunicações. Brasília, 2015. Disponível em: <https://dsic.planalto.gov.br/documentos/publicacoes/4_Estrategia_de_SIC.pdf>. Acesso em: 25 de novembro de 2016.

Fonseca, J. J. S. Metodologia da pesquisa científica. Fortaleza: UEC, 2002. Apostila. Disponível em: <https://books.google.com.br/books?id=oB5x2SChpSEC&printsec=frontcover&hl=pt-BR&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false>. Acesso em: 13 de abril de 2017.

Machado, Tiago Gerard. Metodologia de identificação de nível de maturidade de segurança cibernética em smart grid. *Pontifícia Universidade Católica de Campinas*. Campinas, 2016. Disponível em: <http://www.bdtd.ucb.br/tede/tde_busca/arquivo.php?codArquivo=1429>. Acesso em: 15 de janeiro de 2017.

Mandarino Júnior, Raphael. Segurança e defesa do espaço cibernético brasileiro. Recife, Cubzac, 2010.

Manoel, Sérgio da Silva. Governança de Segurança da Informação: como criar oportunidades para o seu negócio. Rio de Janeiro, Brasport, 2014.

NIST. Framework for Improving Critical Infrastructure Cybersecurity. Version 1.0. Gaithersburg, MD: National Institute of Standards and Technology. 2014. Disponível em: <https://www.nist.gov/sites/default/files/documents/cyberframework/cybersecurity-framework-021214.pdf>. Acesso em: 30 de novembro de 2016.

Rahman, H. A.; Marti, J. R.; Srivastava, K. D. A Hybrid Systems Model to Simulate Cyber Interdependencies between Critical Infrastructures. *International Journal of Critical Infrastructures*,7(4): 265–288. 2011. Disponível em: <http://dx.doi.org/10.1504/IJCIS.2011.045056>. Acesso em 10 fevereiro de 2017.

The President. The President of the United States: Executive Order 13636 Improving Critical Infrastructure Cybersecurity. Federal Register/Presidential Documents, 78(33): February 19, 2013. Washington, DC: U.S. National Archives and Records Administration. Disponível em: <https://www.gpo.gov/fdsys/pkg/FR-2013-02-19/pdf/2013-03915.pdf>. Acesso em: 23 de março de 2017.

U.S. Department of Defense. The Software Engineering Institute. Capability Maturity Model® Integration (CMMI), Version 1.1. Carnegie Mellon University. Disponível em: <http://resources.sei.cmu.edu/asset_files/TechnicalReport/2002_005_001_14042.pdf>. Acesso em: 23 de novembro de 2017.

U.S. Department of Energy. Cybersecurity Capability Maturity Model (C2M2 v1.1). Department of Energy. Washington, DC: U.S. 2014. Disponível em: <https://energy.gov/sites/prod/files/2014/03/f13/C2M2-v1-1_cor.pdf>. Acesso em: 20 de novembro de 2016.

White, Gregory B. The Community Cyber Security Maturity Model (CCSMM). Hawaii International Conference on System Sciences. The Center for Infrastructure Assurance and Security. The University of Texas at San Antonio. 2007. Disponível em: <https://www.computer.org/csdl/proceedings/hicss/2007/2755/00/27550099b.pdf>. Acesso em: 18 de janeiro de 2017.

Xiao-Juan, l.; Li-Zhen, H. Vulnerability and Interdependency of Critical Infrastructure: A Review. *Third International Conference on Infrastructure Systems and Services*: Next Generation Infrastructure Systems for Eco-Cities (INFRA): 1–5. 2010. Disponível em: <http://dx.doi.org/10.1109/INFRA.2010.5679237>. Acesso em 17 janeiro de 2017.

Coleção de dados sobre ataques a dispositivos de Internet das Coisas

Marcos Felipe B. de Abreu[1], **Thierson Couto Rosa**[1], **Kleber Vieira Cardoso**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Alameda Palmeiras, Quadra D, Câmpus Samambaia CEP 74690-900 - Goiânia - GO – Brazil

{marcos, thierson, kleber}@inf.ufg.br

Abstract.

The number of Internet of Things (IoT) devices has increased every day and along with this growth arises the security concerns. Several techniques have been studied for the prevention, detection and treatment of attacks in conventional networks, such as the work of KDD CUP 99 that proposed a labeled collection, which has been quite exploited in recent decades. A good evaluation of techniques and algorithms of intrusion detection systems is related to the existence of good datasets. However, few works exploit the detection of attacks on Internet of Things and until now no collection of data based in network has been proposed for this problem. Along with new technologies and devices arise new techniques of invasion, and even more elaborated. Therefore, it is necessary to treat the attack detection problem in a special way. In view of this, this work is dedicated to setting up a test environment that represents an Internet of Things network, collecting normal device traffic, simulating attacks, assembling a collection of data and analyzing it. For this, we run invasion tests on emulated devices, resulting in a new collection of data. We validate the new collection by applying machine learning algorithms and comparing with the KDD collection.

Resumo. *O número de dispositivos de Internet das Coisas (IoT) vêm aumentado a cada dia e junto com esse crescimento surge as preocupações com a segurança. Diversas técnicas têm sido estudadas para prevenção, detecção e tratamento de ataques em redes convencionais, a exemplo do trabalho do KDD CUP 99 que propôs uma coleção rotulada, a qual vem sendo bastante explorada nas últimas décadas. Uma boa avaliação de técnicas e algoritmos de sistemas de detecção de intrusão está relacionada à existência de boas coleções de dados. Entretanto, poucos trabalhos exploram a detecção de ataques em Internet das Coisas e até então nenhuma coleção de dados baseada na análise da rede foi proposta para o problema. Junto com novas tecnologias e dispositivos surgem novas técnicas de invasão, específicas e mais elaboradas. Logo, é preciso tratar o problema de detecção de ataques em IoT de forma especial. Tendo em vista isso, este trabalho dedica-se a montar um ambiente de teste que represente uma rede de Internet das Coisas, coletar o tráfego normal dos dispositivos, simular ataques, montar uma coleção de dados e fazer uma análise sobre a mesma. Para isso são executados testes de invasão a dispositivos emulados, como resultado obtivemos uma nova coleção de dados. Validamos a nova coleção aplicando algoritmos de aprendizado de máquina e comparando-a com a coleção KDD.*

1. Introdução

Internet das Coisas, do inglês, *Internet of Things* (*IoT*), é um paradigma que está ganhando espaço na área de comunicações sem fio modernas devido ao seu grande impacto na vida da população. A quantidade de dispositivos conectados é enorme, com cerca de 9 bilhões de coisas conectadas à internet em 2013 e há a previsão de que mais de 24 bilhões de "coisas" estejam conectadas até 2020 [J. Gubbi 2013]. E-saúde, assistentes pessoais, vida assistida são exemplos de aplicações presentes em um ambiente doméstico, enquanto no contexto corporativo temos, como exemplos, automação, logística, gerenciamento de negócios/pessoas e transporte inteligente [Gomez and Paradells 2010].

Dispositivos de Internet das Coisas geralmente possuem características bastante comuns como, baixo consumo energético, *hardware* simples e única funcionalidade. O *software* desses dispositivos é embutido em um sistema embarcado em *hardware* e é desenvolvido para ser o mais simples possível. Além das funcionalidades necessárias para aplicação, o fabricante deve se preocupar em garantir a segurança nesses dispositivos. Entretanto, soluções de segurança, em geral, adicionam um custo, tanto computacional quanto financeiro, no produto final.

Atualmente há uma corrida entre os fabricantes para lançarem novos produtos no mercado [Ismail 2018]. Nesse contexto, vários produtos saem de fábrica com defeitos no software, que posteriormente poderão ser explorados tornando-se vulnerabilidades. A vulnerabilidade de um dispositivo geralmente é encontrada em vários outros da mesma linha, o que facilita a escalabilidade de ataques.

Em geral, na defesa contra ataques em uma rede são usadas *Firewalls*, que controlam acessos à rede, e os Sistemas de Detecção de Intrusão, do inglês *Intrusion Detection System* (*IDS*), que analisam o tráfego da rede e criam um padrão que descreve o tráfego normal. A tarefa de aprendizado do *IDS* é construir um modelo preditivo, isto é, um classificador, capaz de distinguir entre conexões "ruins", chamadas de intrusões ou ataques, e conexões normais, "boas". Algoritmos de aprendizado de máquina têm tido bons resultados em problemas de detecção de anomalias.

Uma boa avaliação de técnicas e algoritmos de detecção de intrusão está relacionada à existência de bons conjuntos de dados de ataques. Por duas décadas, a coleção KDD Cup 99 [Stolfo et al. 1999] vem sendo a principal referência de coleção, sendo analisada em diversos trabalhos que envolvem detecção de ataque, embora possua várias críticas quanto a sua montagem. A KDD é uma coleção advinda do processamento do tráfego de rede coletado no trabalho do *Defense Advanced Research Projects Agency* (DARPA), em 1998.

Uma rede *IoT* possui características de tráfego bem distintas das redes convencionais. As coleções de dados públicas disponíveis atualmente não representavam o universo de ataques direcionados a dispositivos de Internet das Coisas. Dada essa necessidade, o objetivo principal deste trabalho é criar um ambiente que represente uma rede *IoT*, realizar experimentos que visam coletar dados do tráfego de rede normal de dispositivos e de ataques, montar uma coleção de dados rotulada e fazer uma análise sobre a mesma.

Atualmente, no contexto de Internet das Coisas, os dispositivos que mais frequentemente são alvos de ataques são as Câmeras IP e os Roteadores [Micro 2017]. O principal motivo é quantidade enorme desses dispositivos que estão conectados à Internet e podem

ser localizados por ferramentas como as do site Shodan [Shodan 2013].

Através de um ambiente de experimentação, foi coletado o tráfego normal e de ataques, direcionados a esses dispositivos. Como resultado obtemos uma coleção de dados que representa uma rede *IoT*. Essa coleção foi validada e comparada com a coleção KDD, através da aplicação de aprendizado de máquina.

O resto do trabalho está organizado em quatro seções. Na Seção 2 são apresentados trabalhos relacionados à detecção de ataques e montagem de coleções de dados. A Seção 3 descreve a metodologia aplicada no trabalho. Na Seção 4, é descrito o processo de montagem da coleção e é feita uma análise aplicando classificadores na mesma. O último capítulo traz as considerações finais e discute as possibilidades de trabalhos futuros.

2. Trabalhos Relacionados

Devido a todos os problemas discutidos na seção anterior, algumas coleções de dados foram propostas nos últimos anos. A principal motivação para criação dessas coleções é prover diferentes tipos de dados, que caracterizam diferentes ambientes de rede. [Maciá-Fernández et al. 2018] propõem uma nova coleção denominada UGR'16. Os dados foram obtidos de uma rede real do fornecedor de acesso à Internet (ISP) Tier 3. O ISP Tier 3 é um provedor de serviços de nuvem e grande parte dos serviços dessa rede são virtualizados. Três classes de ataques foram analisados do trabalho: *Denial of Service*, *Port scanning* e *Botnet traffic*.

O trabalho de [Chen et al. 2016] objetiva avaliar vulnerabilidades em dispositivos com *kernel* baseado em Linux. Para isso são executados ataques sobre *firmwares* emulados de dispositivos. Como resultado, é apresentado um relatório apontando uma série de dispositivos vulneráveis aos ataques analisados. O trabalho mostra que vários dos ataques testados afetam imagens de *firmwares* de mais de um fornecedor, sugerindo que, um mesmo código fonte é compartilhado entre fabricantes de dispositivos.

Alguns trabalhos vão para linha de detecção de *botnets*. *Botnet* é uma rede, controlada por um agente mal intencionado, onde vários dispositivos unem esforços para atingir um objetivo. Os ataques envolvendo *botnets* são precedidos pela invasão de dispositivos, onde o atacante ganha acesso ao sistema e passa a controlá-lo. Quando uma *botnet* entra em ação é mais difícil de contê-la, logo os Sistemas de Detecção de Intrusão têm um papel fundamental, agindo na primeira fase da infecção, caracterizada pela tentativa de ganho de acesso ao dispositivo. [Garcia et al. 2014] propõe uma base de dados para comparação de métodos de detecção de *botnets*. [Gu et al. 2008] utiliza técnicas de correlação entre *clusters* para identificar *hosts* que compartilham padrões semelhantes de atividades maliciosa.

[Bezerra et al. 2018] propõe uma solução de um sistema de detecção de intrusão baseado em *host*. Para isso, ele coleta dados referentes a operação dos dispositivos como uso de CPU e memória com objetivo de aprender o padrão de funcionamento do dispositivo. A desvantagem desta abordagem é que ela não detecta tentativas de invasão nos dispositivos.

Ataques direcionados às camadas de Enlace e Física envolvem os protocolos de comunicação sem fio. Para realizar ataques dessa natureza é necessário estar dentro do raio de comunicação do dispositivo alvo e os objetivos mais comuns são, tornar os recur-

sos indisponíveis ou se passar por um dispositivo. Identificadores únicos são usados na camada de Enlace para diferenciação de dispositivos e controle de acesso. Isso não resolve o problema por que esses identificadores podem ser forjados. Uma solução para isso, é analisar o sinal que um dispositivo envia e extrair características que o diferencie de outros. O processo de fabricação das placas que geram o sinal adiciona pequenas imperfeições, não propositais, na placa, o que influencia no sinal gerado. Essas imperfeições não prejudicam a comunicação, mas são suficientes para diferenciar dispositivos. [Wang et al. 2016] analisa a desempenho de um classificador que atua na camada física coletando características do sinal e diferenciando dispositivos. Para isso são coletados dados de 40 placas. Trabalhos nessa área têm obtido altas acurácias na classificação.

3. Metodologia

Na literatura poucos trabalhos se dedicaram a montar coleção de dados para avaliação de Sistemas de Detecção de Intrusão. A principal razão dessa lacuna está na natureza dos dados coletados em uma rede real. Os dados podem conter informações sensíveis como senhas, segredos de negócios e padrões de acesso de usuários [Sommer and Paxson 2010]. Tendo em vista essa dificuldade, pesquisadores propuseram como alternativa a virtualização e a simulação. Toda coleta de dados do DARPA, que deu origem a coleção de dados KDD, foi realizada em um ambiente virtualizado. A Figura 1 apresenta o fluxo da metodologia adotada neste trabalho.

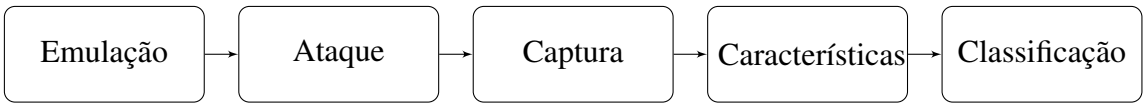


Figura 1. Fluxo dos experimentos

O primeiro passo no fluxo dos experimentos é montar o ambiente para realização dos testes. O grande desafio em criar um ambiente de teste é montar uma rede com dispositivos que descrevam um cenário de Internet das Coisas. O uso de dispositivos físicos na realização de experimentos é um tanto inviável quando se pensa em escala. A solução para esse problema é a criação de um ambiente virtual que reporte bem um ambiente real. O tráfego capturado de um ataque representa os dados enviados do atacante e a resposta do dispositivo a esses dados. A partir desses dados são extraídas as características necessárias para a classificação.

As próximas seções descrevem em detalhes cada passo da metodologia e as ferramentas utilizadas para montagem da coleção.

3.1. Ambiente de testes

Emular um dispositivo físico é um processo um tanto complicado. O sistema encontrado nesses dispositivos é projetado para executar em hardware de arquitetura específica. Para emular os dispositivos usados nos experimentos foi utilizada a ferramenta *Firmadyne* apresentada em [Chen et al. 2016]. *Firmadyne* é um sistema que integra várias ferramentas para emular dispositivos baseados no kernel do sistema operacional Linux. Na base da emulação do Firmadyne temos a figura do *QEMU* [Qemu 2018]. O *QEMU* é um dos mais famosos virtualizadores de sistemas baseados em Linux. O processo completo de

emulação consiste em extrair a imagem do firmware, montar o sistema de arquivos e configurar uma interface de rede para a interação entre o *firmware* e a máquina ao qual ele foi emulado. Após a emulação do dispositivo, o *QUMU* oferece um terminal, onde é possível interagir com o sistema.

O ambiente de testes foi montado em um servidor local com máquinas virtuais (VMs) instaladas. A Figura 2 descreve a arquitetura do ambiente de testes.

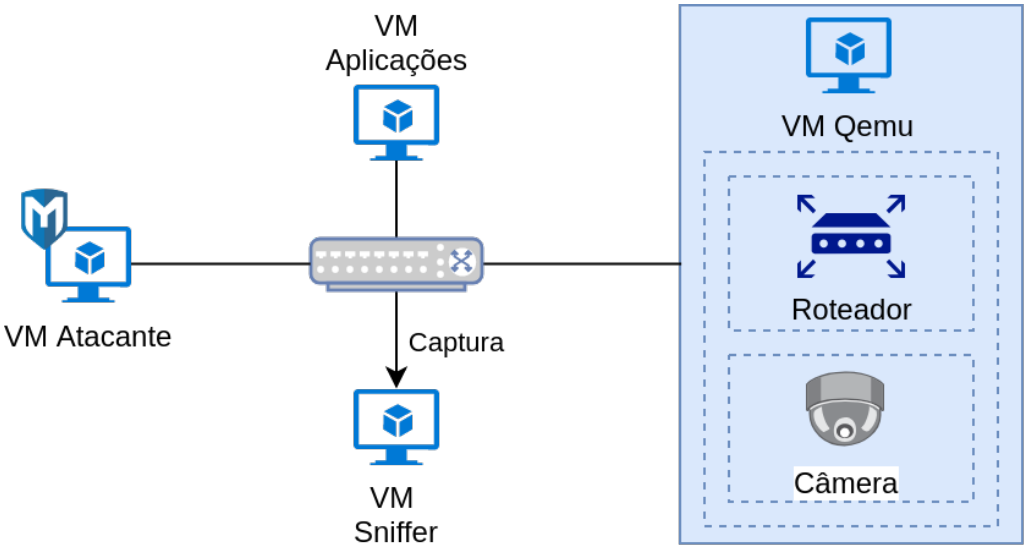


Figura 2. Topologia utilizada para a realização dos experimentos

As imagens dos sistemas dos dispositivos são encontradas nos sites dos próprios fabricantes e possuem a extensão ".img" ou ".bin". Elas são disponibilizadas com intuito de fornecer aos usuários a possibilidade de restaurarem ou atualizarem o sistema. Alguns fabricantes preferem não disponibilizar as imagens publicamente em sites, sendo necessário contatar o suporte para obter as mesmas.

Diversos erros podem ocorrer no processo de emulação do *firmware* e nem todas as imagens obtidas dos dispositivos podem ser emuladas. Os problemas mais comuns são falhas na montagem do sistema de arquivos e erros no carregamento do sistema operacional. Esse último ocorre devido ao fato de que partes do sistema operacional precisam de se comunicar com algum componente de hardware.

Para capturar o trafego gerado pela execução dos ataques foi utilizada a ferramenta *tcpdump* [tcpdump 2017]. O *tcpdump* é uma ferramenta utilizada para monitorar os pacotes trafegados numa rede de computadores. Ele captura o cabeçalho e o conteúdo dos pacotes que passam por uma interface de rede selecionada e é possível salvar essas capturas em arquivos, que podem ser abertos por ferramentas de análise como o *Wireshark*. É importante ressaltar que para deixar o ambiente mais realista foi adicionado um *delay* nas interfaces de rede visto que, como se trata de uma comunicação entre VMS, o atraso dos pacotes é quase zero, o que influência nas características coletadas da rede.

3.2. Dispositivos

O conceito de Internet das Coisas cobre uma quantidade enorme de aplicações, envolvendo diversos dispositivos e diferentes cenários. Uma solução de segurança para *IoT*

deve ser generalista, não deixando de considerar essa heterogeneidade de dispositivos. Duas categorias de dispositivos foram consideradas neste trabalho, câmeras IP e roteadores WiFi. A principal motivação para escolha de roteadores e câmeras vem do atual cenário de Segurança da Informação. Esses dispositivos foram os principais alvos dos ataques nos últimos anos, sendo um dos maiores o *Dyn cyberattack* [Kaspersky 2016], em 2016. Além disso, roteadores e câmeras IP representam bem a arquitetura de um dispositivo de Internet das Coisas tanto em software como em hardware.

As câmeras IP, no contexto de Internet das Coisas, atuam como sensores que possibilitam o monitoramento de ambientes, tendo como aplicação principal a segurança pública e privada. Outros exemplos de aplicação de Câmeras são no monitoramento de plantações em fazendas inteligentes (*Smart Farms*) e na detecção de faces para o controle de acesso.

Roteadores também têm um papel importante em Internet das Coisas. Vários deles atuam como *gateways*, fazendo a ponte entre dispositivos e a Internet Pública. Em apenas um roteador podem estar conectados um número grande de "coisas", justificando assim a preocupação com a segurança dos mesmos.

3.3. Perfil do trafego normal

Como descrito anteriormente, no trabalho avaliamos duas categorias de dispositivos, Câmeras IP e Roteadores. O perfil de tráfego normal em um roteador está relacionado às configurações disponíveis em sua interface web. Vale ressaltar que, não estamos avaliando o tráfego dos dispositivos conectados no roteador, mas sim o tráfego direcionadas a ele. Este tráfego é caracterizado pela interação do usuário com a interface web do roteador. A interface web fornece ao usuário a capacidade de efetuar configurações como, mudanças de senha, alterações de *DNS*, abertura de portas, detre outras. Ao emular um dispositivo desta classe, conseguimos acesso à sua interface web, sendo possível simular a interação de um usuário com o mesmo.

Algumas funcionalidades dos dispositivos não estão disponíveis na emulação dos mesmos. Estas funcionalidades, em geral, são dependentes de componentes de hardware específicos, como sensores e atuadores, que são verificados no processo de inicialização dos dispositivos. Ao emular uma câmera IP, perdemos a funcionalidade de coleta e transmissão de imagens, sendo necessário simular isso de outra forma. O tráfego normal da câmera foi gerado por simulação através de uma ferramenta disponível em [Singh 2014]. O simulador gera dados de diversas aplicações de *IoT*, incluindo transmissão de video de câmeras IP. Nos simulador, é possível definir padrões de transmissão como, taxa de quadros por segundo e resolução da câmera.

3.4. Execução dos ataques

Um *exploit* (exploração) é o meio pelo qual um invasor aproveita de uma falha em um sistema, um aplicativo ou um serviço. O *exploit* é usado para atacar um sistema de uma maneira que o invasor obtém um resultado específico, que o desenvolvedor nunca pretendeu. As explorações comuns incluem estouros de *buffer*, vulnerabilidades de aplicativos da web (como injeção de SQL) e erros de configuração. O *exploit* é basicamente um código que interage com o dispositivo, modificando ou extraindo dados, que serão usados posteriormente para obter privilégios no sistema.

Códigos de ataques são disponibilizados publicamente na internet. Esses códigos são disponibilizados com intuito de auxiliarem pesquisadores e profissionais de segurança da informação no estudo e desenvolvimento de ferramentas de mitigação de ataques. Ao surgir um novo ataque, espera-se que o fabricante disponibilize rapidamente uma correção para a vulnerabilidade do dispositivo.

Existem ferramentas que automatizam todo o processo de realização de um ataque. Para execução dos ataques direcionados aos dispositivos de Internet das Coisas foi utilizado o Metasploit [Maynor 2011]. Metasploit é uma ferramenta de código aberto que tem objetivo de fornecer à comunidade um ambiente para verificação de vulnerabilidades.

O *framework* Metasploit foi desenvolvido com a intenção de tornar a vida de especialistas de segurança mais fácil. Os usuários da ferramenta foram considerados ser profissionais de segurança de rede, administradores de rede, fornecedores de produtos e outros pesquisadores de segurança. Cada um usaria a ferramenta dentro das diretrizes de sua própria disciplina, profissionais de segurança para testes de penetração, fornecedores para testes de seus produtos e outros pesquisadores de segurança para talvez o desenvolvimento de outros *exploits* [Maynor 2011].

O Metasploit possuí um banco de dados de códigos de ataques, do qual é possível selecionar um *exploit* específico, cabendo ele automatizar sua execução. Esse banco de dados é construído de forma colaborativa e para cada nova vulnerabilidade descoberta um modulo é criado. Atualmente 1821 *exploits* estão disponíveis para uso.

Ao selecionar um ataque no banco de dados é possível configurar alguns parâmetros, obrigatórios ou não. Na maioria dos casos, o único parâmetro obrigatório é o *host* alvo (*RHOST*), sendo os demais opcionais ou específicos de uma determinado *exploit*.

De um total de 1821 *exploits* disponíveis no Metasploit, selecionamos 68, cujo critério de escolha foi o de estarem relacionados com algum dispositivo de Internet das Coisas. Cada exploit é direcionado a um conjunto específico de dispositivos, denominados vulneráveis. O atacante nem sempre tem alvos específicos e, na maioria das vezes, os ataques são executados por *scripts* que automatizam o processo, com objetivo de atingir um número maior de dispositivos. Portanto, neste trabalho é considerado tanto dispositivos vulneráveis ou não a um determinado ataque. Vale ressaltar que, a execução de um *exploit* pode ter comportamentos diferentes ao interagir com dispositivos não vulneráveis. Por exemplo, no fluxo de execução do *script* pode conter uma verificação se um determinado serviço está executando ou não, modificando o comportamento do *exploit* de acordo com a resposta.

3.5. Descrição dos ataques

Os ataques executados para montagem da coleção encontram-se em três categorias: execução de código remoto, desvio de autenticação e scanner de vulnerabilidades. Abaixo segue uma descrição dos mesmas.

3.5.1. Execução de código remoto

É um ataque em que um invasor executa códigos arbitrários em uma máquina de destino. Geralmente ele é usado para explorar um *bug* em alguma parte do sistema. A exploração de um *bug* pode permitir que um atacante assuma completamente o processo vulnerável. A partir daí, o invasor pode assumir controle total sobre a máquina em que o processo está sendo executado [wikipedia 2018].

A execução de um código é comumente alcançado através do controle sobre o ponteiro de instrução de um processo em execução. O ponteiro de instrução aponta para a próxima instrução no processo que será executado. O controle sobre o valor do ponteiro de instrução, portanto, dá controle sobre qual instrução é executada a seguir. Para executar código remoto, muitos *exploits* injetam código no processo (por exemplo, enviando entrada para ele que é armazenado em um *buffer* de entrada na memória RAM) e use uma vulnerabilidade para alterar o ponteiro de instrução para que ele aponte para o código injetado. O código injetado será executado automaticamente. Essa categoria de ataque explora o fato de que a maioria dos computadores não faz uma distinção geral entre código e dados, de modo que o código mal-intencionado pode ser camuflado como dados de entrada inofensivos.

Uma vez que o invasor executa um código arbitrário diretamente no sistema operacional, geralmente há uma tentativa de exploração de escalonamento de privilégios para obter controle adicional. Com ou sem esse controle aprimorado, as explorações têm o potencial de causar danos graves ou obter o controle sobre o dispositivo *IoT*.

3.5.2. Desvio de autenticação

È um ataque explora a negligência de fabricantes ou presença de *bugs* que permitam que páginas de autenticação sejam ignoradas e o sistema seja acessado diretamente sem nenhuma verificação.

3.5.3. Scanner de vulnerabilidades

É uma tentativa de ganhar informação sobre dispositivos conectados em uma rede, usando ferramentas de mapeamento de rede. A partir dessas informações o atacante pode planejar ataques direcionados para os dispositivos específicos. Geralmente é o primeiro passo de uma sequência de ataques. *Scanner* de vulnerabilidades não é um ataque, por definição, mas pode ser considerado por que é um grande indicativo do início de um ataque. Logo, é ideal que ferramentas de detecção de intrusão capture esses eventos visto que, existem técnicas que tentam mascarar a execução do *scanner*.

3.6. Características usadas na coleção KDD Cup 99

A escolha das características é uma parte fundamental no trabalho de classificação. As características usadas pela KDD no problema de detecção de ataques são derivadas do trabalho de [Stolfo et al. 2000]. Esse trabalho se preocupa em analisar e definir características que são relevantes para problemas de detecção de anomalias. Aplicando algorit-

mos de regras de associação, o trabalho selecionou 41 características, derivadas dos dados brutos coletados da rede, divididas nas seguintes categorias:

Características baseadas no ”mesmo host”: examinam apenas as conexões nos últimos dois segundos que possuem o mesmo host de destino que a conexão atual e calculam as estatísticas relacionadas ao comportamento do protocolo, serviço, etc. Exemplos dessas caraterísticas são: Duração da conexão, tipo do protocolo e número de bytes de dados trocados entre a origem destino do tráfego.

As características não estão restritas somente ao nível de pacotes, mas se estendem para camada de aplicação. [Stolfo et al. 2000] usou o conhecimento de domínio para adicionar recursos que procuram comportamento suspeito nas partes de dados, como o número de tentativas de login com falha. Esses recursos são chamados de recursos de conteúdo (“content”). Essas características não se encontram disponíveis em nossa coleção pois derivam-se de um processamento específico da camada de aplicação.

Características semelhantes do “mesmo serviço”: examinam apenas as conexões nos últimos dois segundos que possuem o mesmo serviço que a conexão atual. Características “mesmo host” e “mesmo serviço”: são chamados de recursos de tráfego baseados em tempo dos registros de conexão. Exemplos dessas características são: Porcentagem de conexões que tiveram erros SYN, número de conexões para o mesmo serviço da conexão atual e porcentagem de conexões para host diferente.

3.7. Extração das características

Após a captura, os dados são salvos em arquivos. Cada arquivo representa o tráfego da rede e é rotulado como sendo de tráfego normal ou anômalo, de acordo com o experimento. A partir desses arquivos, que se encontram no formato pcap, é necessário extrair as características que serão usadas na classificação. As características analisadas no trabalho são as mesmas 41 usadas na coleção KDD e a Figura 3 apresenta algumas delas.

[0,tcp,telnet,S3,0,44,0,0, ... ,0.00,0.00,0.00,255,79,0.31,anomaly]

[0,udp,private,SF,53,55,0, ... ,255,1.00,0.00,0.87,00,0.00,normal]

Figura 3. Exemplo de uma tupla da coleção

Para realização do processo de extração das características a partir dos pacotes, foi utilizada uma ferramenta elaborada em um projeto na University of Bergen, que se encontra disponível em [kdd 2015]. O autor da ferramenta alerta que algumas características podem não ser calculadas exatamente da mesma maneira da coleção KDD, pois, não há documentação explicando com detalhes a implementação usada para montagem da KDD. Os algoritmos são baseados nos artigos existentes e na observações de valores da coleção. Por fim, a rotulagem dos dados, ou seja, a associação entre a tupla e o rótulo (normal ou anomalia) é feita manualmente após a extração das características pela ferramenta.

4. Experimentos e Avaliação

Conforme a metodologia descrita no capítulo anterior, coletamos os pacotes do tráfego de rede dispositivos, tanto normal quanto de ataque. Os dados foram coletados a partir

de quatro dispositivos emulados, apresentados na Tabela 1. O critério de escolha do modelo e da versão dos dispositivos foi eles serem vulneráveis a alguma categoria de ataque realizada.

Dispositivo	Tipo	Versão do firmware
AirTies Airs5650	Roteador wireless	Airs5650v3
D-Link DCS-930L	Câmera IP	dcs930lb1_v2.01.03
Netgear dgn1000b	Roteador wireless	DGN1000_V1.1.00.56_NA
TP-Link SC2020n	Câmera IP	TL-SC2020N V1

Tabela 1. Dispositivos utilizados nos experimentos

Após o processamento dos dados brutos e da extração das características, como resultado obtivemos 220 tuplas, rotuladas como "ataque" e 960 tuplas rotuladas como "normal", totalizando 1180 tuplas. A proporção de 18,6% de ataques se assemelha aos 19,8% da coleção KDD.

Na análise usamos três classificadores, Multilayer Perceptron, Naive Bayes e Random Forest. Fizemos dois experimentos, um envolvendo a coleção NSL KDD e outro a coleção proposta.

Aplicamos a técnica de validação cruzada nos experimentos que envolveram nossa coleção. A validação cruzada é uma técnica que avalia a capacidade de generalização de um modelo, a partir de uma coleção de dados [Kohavi et al. 1995]. Esta técnica é amplamente empregada em problemas onde o objetivo da modelagem é a predição. Para isso, a coleção completa é dividida em conjuntos menores e em cada rodada é selecionado uma porção dos dados para treinamento e a outra parte para a verificação. Dividimos a coleção em 10 partes iguais sendo que, em cada rodada uma dessas partes é usada para testar o modelo e as restantes para treinamento. A base NSL KDD já está dividida entre treinamento e teste.

No primeiro experimento analisamos a coleção NSL KDD. Os resultados da classificação são semelhantes aos obtidos em [Tavallaee et al. 2009]. O classificador que obteve a melhor performance (80,4%) foi o Random Forest, seguido do Naive Bayes (76,1%) e Multilayer Perceptron (75,8%). A Figura 4 mostra os resultados da avaliação dos algoritmos em termos de acurácia.

No segundo experimento avaliamos nossa coleção aplicando os classificadores nela. A acurácia nesses testes foi melhor que o do primeiro experimento. O Random Forest obteve a melhor performance nesse experimento, seguido pelo Multilayer Perceptron (97,4%) e O Naive Bayes (87,1%).

O tráfego de dados em uma rede convencional tende a ser aleatório devido à multiplicidade de aplicações, o que pode tornar o problema de detectar anomalias mais complexo . O comportamento dos usuários da rede pode mudar conforme o tempo, o que dificulta a caracterização do tráfego normal. Entretanto, o tráfego em uma rede de Internet das Coisas tende a ser mais previsível, pois, os dispositivos possuem aplicações e comportamentos específicos, similar ao problema de se detectar fraudes em transações bancárias [Sommer and Paxson 2010].

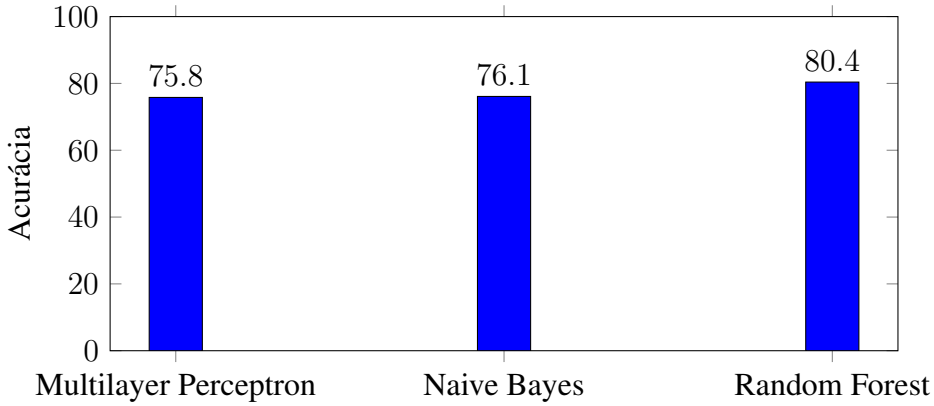


Figura 4. Performance dos algoritmos aplicados na coleção KDD

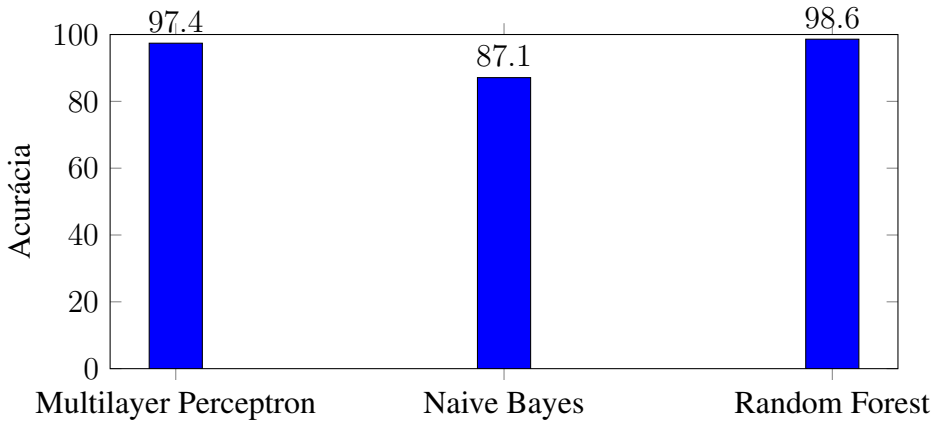


Figura 5. Performance dos algoritmos aplicados na nova coleção

5. Conclusão e Trabalhos Futuros

O trabalho apresentou uma metodologia para montagem de uma coleção de dados de classificação de ataques de uma rede de Internet das Coisas. A partir de um ambiente virtualizado foi possível emular diversos dispositivos. Realizando experimentos e coletando dados do tráfego da rede, montamos uma coleção rotulada com as mesmas características da coleção KDD Cup 99.

Avaliamos a Coleção aplicando algoritmos de aprendizado de máquina e comparando os resultados com a coleção NSL KDD. O resultado obtido na classificação da nossa coleção foi melhor que a KDD.

Como trabalho futuro pretendemos realizar novos experimentos, integrando dispositivos de Internet das Coisas reais com nosso ambiente virtual, assim aproximando mais do tráfego real e promovendo o aumento da nossa coleção. Também pretendemos avaliar e propor novas características para o problema de detecção de intrusão em uma rede de Internet das Coisas. Para isso, os dados brutos (pacotes) podem ser analisados de modo a encontrar padrões que melhor descrevam um ataque.

A coleção gerada a partir deste trabalho e mais detalhes sobre performance dos algoritmos de classificação estão disponíveis em [Abreu 2018].

Referências

(2015). kdd99 feature extractor. https://github.com/AI-IDS/kdd99_feature_extractor. Acesso: 2019-01-03.

Abreu, M. (2018). Idsiot. https://github.com/marcosfelipp/IDSIoT. Acesso: 2019-03-08.

Bezerra, V. H., da Costa, V. G. T., Martins, R. A., Junior, S. B., Miani, R. S., and Zarpelao, B. B. (2018). Providing iot host-based datasets for intrusion detection research. In *SBSeg 2018*, pages 15–28. SBC.

Chen, D. D., Woo, M., Brumley, D., and Egele, M. (2016). Towards automated dynamic analysis for linux-based embedded firmware. In *NDSS*.

Garcia, S., Grill, M., Stiborek, J., and Zunino, A. (2014). An empirical comparison of botnet detection methods. *computers & security*, 45:100–123.

Gomez, C. and Paradells, J. (2010). Wireless home automation networks: A survey of architectures and technologies. volume 48, pages 92–101. IEEE.

Gu, G., Perdisci, R., Zhang, J., and Lee, W. (2008). Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection.

Ismail, N. (2018). The internet of things: The security crisis of 2018? https://www.information-age.com/internet-things-security-crisis-123470475/. Acesso: 2019-01-03.

J. Gubbi, R. Buyya, S. M. M. P. (2013). Internet of things (iot): A vision, architectural elements, and future directions in future generation computer systems. volume 29, pages 1645–1660.

Kaspersky (2016). 2016 dyn cyberattack. https://www.kaspersky.com/blog/attack-on-dyn-explained/13325/. Acesso: 2018-10-14.

Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. volume 14, pages 1137–1145. Montreal, Canada.

Maciá-Fernández, G., Camacho, J., Magán-Carrión, R., García-Teodoro, P., and Therón, R. (2018). Ugr '16: A new dataset for the evaluation of cyclostationarity-based network idss. *Computers & Security*, 73:411–424.

Maynor, D. (2011). *Metasploit toolkit for penetration testing, exploit development, and vulnerability research*. Elsevier.

Micro, T. (2017). Securing your routers. https://www.trendmicro.com/vinfo/ie/security/news/internet-of-things/securing-routers-against-mirai-home-network-attacks. Acesso: 2019-01-10.

Qemu (2018). Qemu v3.1. https://qemu.weilnetz.de/doc/qemu-doc.html. Acesso: 2018-11-20.

Shodan (2013). Shodan. https://www.shodan.io/. Acesso: 2019-01-10.

Singh, V. (2014). Sensor traffic generator. https://github.com/vr000m/SensorTrafficGenerator. Acesso: 2018-11-16.

Sommer, R. and Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 305–316. IEEE.

Stolfo, J., Fan, W., Lee, W., Prodromidis, A., and Chan, P. K. (2000). Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection. *Results from the JAM Project by Salvatore*, pages 1–15.

Stolfo, S. et al. (1999). Kdd cup 1999 dataset. *UCI KDD repository. http://kdd. ics. uci. edu*.

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pages 1–6. IEEE.

tcpdump (2017). tcpdump version: 4.9.2. `http://www.tcpdump.org/`. Acesso: 2019-01-10.

Wang, W., Sun, Z., Ren, K., and Zhu, B. (2016). Increasing user capacity of wireless physical-layer identification in internet of things. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, pages 1–6. IEEE.

wikipedia (2018). Arbitrary code execution. `https://en.wikipedia.org/wiki/Arbitrary_code_execution`. Acesso: 2018-10-14.

Uma Taxonomia para Descrever e Caracterizar Estratégias de Mitigação de Ataques DDoS em Ambientes IoT Usando Tecnologias SDN

Esau Silva[1], **Felipe S. Dantas Silva** [1,2], **Marcilio O. O. Lemos**[1,2], **Augusto Venâncio Neto** [2,3]

[1]LaTARC Research Lab
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN)
Natal-RN, Brasil

[2]Departamento de Informática e Matemática Aplicada (DIMAp)
Universidade Federal do Rio Grande do Norte (UFRN) – Natal-RN, Brasil

[3]Instituto de Telecomunicações (IT), Aveiro, Portugal

esau.silva@academico.ifrn.edu.br, felipe.dantas@ifrn.edu.br
marcilio.cc.lemos@gmail.com, augusto@dimap.ufrn.br

Abstract. *The Internet of Things has attracted significant attention from the Information and Communication Technology (ICT) community through the perspective of everyday objects collecting data from the environment and transmitting them via the Internet for post-processing, without the need for human-computer interaction . Although IoT represents a powerful platform for creating new products and services that will benefit a wide variety of verticals industries (e-health, V2X, smart homes, etc.), there are a number of security concerns that need be remedied for their proper implementation, such as vulnerabilities exploited by malicious agents to take control of devices and deliver DDoS attacks on a large scale. Therefore, research was initiated to develop solutions for the detection and containment of IoT-based DDoS attacks, making use of emerging technologies such as the Software Defined Networks (SDN) paradigm. In this sense, the present work presents a taxonomy to describe and characterize the body of SDN solutions against DDoS attacks in IoT scenarios.*

Resumo. *A Internet das Coisas têm atraído significativa atenção da comunidade de Tecnologia da Informação e Comunicação (TIC) pela perspectiva de objetos do cotidiano coletando dados do ambiente e transmitindo-os através da Internet para processamento a posteriori, sem que haja necessidade de interação humano-computador. Embora a IoT represente uma poderosa plataforma para criação de novos produtos e serviços que irão beneficiar uma ampla variedade de verticais (e-health, V2X, smart homes, etc), existem diversas preocupações de segurança que precisam ser remediadas para sua implementação adequada, como as vulnerabilidades explorados por agentes maliciosos para assumir o controle de dispositivos e desferir ataques DDoS em larga escala. Por conseguinte, pesquisas foram iniciadas para desenvolver soluções visando a detecção e contenção de ataques DDoS baseados em IoT, fazendo uso de tecnologias emergentes como o paradigma de Redes Definidas*

por Software (SDN). Neste sentido, o presente trabalho apresenta uma taxonomia para descrever e caracterizar o corpo de soluções SDN contra ataques DDoS em cenários da IoT.

1. Introdução

A Internet das Coisas (IoT, do inglês *Internet of Things*) [Al-Fuqaha et al. 2015] promete revolucionar a ideia que as pessoas têm dos computadores em seu dia a dia, transformando dispositivos em elementos onipresentes que realizam computação de maneira transparente para seus usuários. Por meio do paradigma IoT, a entrega de serviços fornecerá níveis de comodidade sem precedentes, mas que também adicionam novas preocupações de segurança que podem ofuscar seus benefícios [Noor and Hassan 2019].

Na maioria dos cenários IoT, dispositivos interagem com aplicações que são executadas remotamente na rede, o que possibilita a ação de agentes maliciosos que podem assumir o controle dos dispositivos. Desta maneira, é possível haver a interrupção de serviços ou a utilização de dispositivos como ponto de lançamento de ataques para domínios diversos. De fato, o aumento explosivo no tamanho dos Ataques de Negação de Serviço Distribuído (DDoS, do inglês *Distributed Denial of Service*) [Zargar et al. 2013] observados nos últimos anos é frequentemente associado a popularização de aplicações impulsionadas pela IoT. Muitos desses ataques foram bem-sucedidos em interromper serviços básicos da Internet, como o DNS, afetando milhões de usuários ao redor do mundo [Marzano et al. 2018], e gerando severas perdas financeiras as organizações que mantêm esses serviços.

Naturalmente, este cenário criou uma nova demanda por tecnologias que forneçam maneiras mais eficientes de proteger a integridade de serviços de rede frente a ataques DDoS originados por dispositivos IoT. Neste contexto, a relativa facilidade na execução de ataques DDoS em larga escala aumenta a preocupação com segurança ao passo da proliferação das atividades das redes zumbis (*botnets*) [Bertino and Islam 2017]. Um exemplo dentro deste contexto é a *botnet* Mirai [Kolias et al. 2017] [Marzano et al. 2018], que em 2016 indisponibilizou grande parte do acesso à Internet em diversos países. Frente a isso, operadores de rede e pesquisadores dedicaram diversos esforços para endossar níveis adequados de segurança em infraestruturas IoT para impedir que atividades deste tipo sejam realizadas.

Nesse contexto, o advento de tecnologias como as Redes Definidas por Software (SDN, do inglês *Software-Defined Networking*) [Koufopavlou 2015] é considerada pela comunidade de pesquisa como um dos principais avanços tecnológicos na área de redes pela perspectiva de flexibilidade e programabilidade das infraestruturas. A tecnologia SDN envolve o uso de substratos definidos por software, que desacoplam o plano de controle do plano de dados, permitindo assim que funções antes executadas em hardware, pelos nós da infraestrutura de rede, agora passem a ser executadas ao nível de software hospedado em controladores centralizados. Além disso, APIs comuns implementadas tanto pelos controladores SDN quanto pelos nós de rede fornecem a capacidade de (re)programação do sistema em tempo de execução. Nessa abordagem centralizada, controladores SDN podem supervisionar e atuar na infraestrutura, tendo como apoio bases de conhecimento acerca de todo sistema adjacente. Assim, é possível ter acesso, por meio de uma visão holística, a todo a infraestrutura da rede, incluindo os serviços e aplicações,

maximizando as perspectivas de detecção e contenção de ameaças.

Inúmeros estudos adotaram SDN como meio de implementar níveis de segurança nas infraestruturas de rede em diversos segmentos, tais como controle de acesso [Yakasai and Guy 2015], detecção de malwares [Ceron et al. 2016], comunicação confiável de dados [Shi et al. 2017], dentre outros. Recentemente, várias propostas também fizeram uso da abordagem SDN para fornecer proteção às infraestruturas diante das iminentes ameaças causadas por ataques DDoS [Kalkan et al. 2017]. Com esta motivação novas soluções foram empregadas com o intuito de avançar as atuais estratégias para modelos com capacidades avançadas que fornecessem aos operadores de rede a flexibilidade necessária para aprimorar suas defesas diante da evolução dos ataques.

Mesmo SDN tendo sido amplamente empregada no desenvolvimento de novos algoritmos, soluções e estratégias de detecção e contenção de DDoS para serem aplicados de forma eficiente na proteção das infraestruturas, poucos estudos se comprometeram em fornecer uma visão abrangente de sua aplicação no cenário IoT. Isso se faz necessário à medida em que existe uma grande evolução na variação e ação dos atacantes, bem como nos efeitos causados pelas ameaças, de modo que os desenvolvedores possam guiar-se durante a concepção de novas ferramentas. Em levantamento recente do estado da arte, pode-se identificar que poucos trabalhos se comprometeram a fornecer uma visão abrangente das principais soluções propostas para mitigação de ataques DDoS em ambientes IoT através do uso da tecnologia SDN. Esta lacuna é a principal motivação deste estudo uma vez que se faz necessário identificar as principais abordagens propostas neste meio, de modo que se possa compreender claramente suas vantagens e desvantagens e reconhecer as ações esperadas por mecanismos de defesa diante de determinadas situações de ameças impostas pela variabilidade dos ambientes IoT. Em função disso, este trabalho introduz uma taxonomia para descrever e caracterizar as estratégias de mitigação de ataques DDoS que fazem uso da tecnologia SDN, em ambientes IoT.

As principais contribuições fornecidas neste trabalho podem ser sumarizadas da seguinte maneira: sólida classificação das estratégias de mitigação DDoS usando SDN sob diferentes perspectivas, tais como o local de implantação, limitações, estratégia de detecção e mitigação adotada, bem como abrangência na quantidade de soluções revisadas e escopo (diferente dos trabalhos relacionados que concentram-se apenas nos aspectos de segurança em ambientes IoT).

O restante deste trabalho está organizado da seguinte maneira: a Seção 2 aponta os trabalhos relacionados ao tema de pesquisa descrito; a Seção 3 detalha a metodologia adotada na condução desta taxonomia; a Seção 4 apresenta a taxonomia das soluções estudadas; a Seção 5 apresenta a classificação das estratégias revisadas;Por fim na Seção 6 são apresentada as considerações finais e os direcionamentos para trabalhos futuros.

2. Trabalhos Relacionados

Atualmente existem na literatura diversos trabalhos com a proposta de revisar soluções de segurança para diversos problemas em ambientes IoT, como confidencialidade, integridade, controle de acesso, disponibilidade etc. Neste sentido [Kouicem et al. 2018] desenvolveu um estudo sobre os principais desafios de segurança inerentes a importantes áreas de aplicação IoT, como *e-health*, *smart grids*, *smart cities*, dentre outros. Além disso os autores levantaram e classificaram diversas soluções de segurança nesse contexto,

levando em consideração principalmente *(i)* se a solução é baseada em tecnologias emergentes (ex. SDN e *blockchain*) ou tradicionais e; *(ii)* segmento do problema de segurança que a solução aborda.

Já o trabalho desenvolvido em [Kanagavelu and Aung 2019] avalia inicialmente a aplicabilidade do paradigma SDN no desenvolvimento de soluções de segurança para IoT. Além disso os autores revisam diversas estratégias de segurança nesse contexto com base em três categorias principais: *(*i) arquiteturas IoT baseadas em SDN; *ii* arquiteturas IoT baseadas em virtualização e SDN; *iii* mitigação de ataques DDoS em IoT baseado em SDN. Neste sentido [Cherian and Chatterjee 2019], [Kalkan and Zeadally 2018] e [Farris et al. 2019] seguiram a tendência dos trabalhos citados anteriormente, classificando os principais problemas de segurança tomando por referência as camadas da infraestrutura IoT e discutindo diferentes soluções que exploram paradigmas emergentes como virtualização de funções de rede (NFV, do inglês *Network Function Virtualization*), aprendizado de máquina (*machine learning*), blockchain e SDN.

Em [Kalkan and Zeadally 2018] os autores apresentaram uma classificação das soluções com base nas seguintes categorias: *(i)* soluções baseadas em rede; *(ii)* soluções baseadas em tráfego e; *(iii)* soluções baseadas em criptografia, analisando suas vantagens e desvantagens. Em contraste, [Farris et al. 2019] concentrou-se em analisar detalhadamente mecanismos de defesa baseados nas tecnologias SDN e NFV.

Seguindo uma perspectiva diferente dos trabalhos citados anteriormente, [Lohachab and Karambir 2018] se empenhou em criar uma taxonomia dos principais tipos de ataques DDoS e suas variações, avaliando o impacto desses ataques em ambientes IoT e revisando diversos mecanismos de defesa, destacando suas principais vantagens e desvantagens. A Tabela 1 sumariza os trabalhos relacionados aqui apresentados, diferenciando-os pelo escopo, quantidade de soluções de mitigação revisadas e sob o aspecto da classificação da mitigação.

Trabalho	**Escopo**	**Quantidade de soluções de mitigação SDN revisadas**	**Classifica as soluções de mitigação?**
[Kouicem et al. 2018]	Segurança em IoT	2	✗
[Kalkan and Zeadally 2018]	Segurança em IoT	1	✗
[Lohachab and Karambir 2018]	Segurança em IoT	1	✗
[Kanagavelu and Aung 2019]	Segurança em IoT	2	✗
[Cherian and Chatterjee 2019]	Segurança em IoT	4	✗
[Farris et al. 2019]	Segurança em IoT	4	✗
Proposta Atual	**Ataques DDoS em IoT**	**9**	✓

Tabela 1. Comparativo entre os trabalhos relacionados

Apesar das contribuições oferecidas pelos trabalhos de revisão citados anteriormente, em apresentar diversas soluções de mitigação, nenhum deles propõe uma revisão sólida, quando se trata de mecanismos de mitigação de ataques DDoS usando o paradigma SDN em redes IoT, capaz de classificar apropriadamente as soluções existentes. Isso se da principalmente pelo fato dos estudos encontrados se limitarem a abordar com pouca profundidade o assunto em questão. Por essa razão este trabalho tem como proposta definir uma taxonomia compreensível das soluções de mitigação no contexto IoT fazendo uso de tecnologias SDN, com a finalidade de fornecer insumos a pesquisadores interessados

no desenvolvimento de novos mecanismos de mitigação.

3. Taxonomia das soluções de mitigação de ataques DDoS

Para conceber uma taxonomia compreensível, neste estudo foram observadas algumas das principais características das soluções de mitigação de ataques DDoS levantadas. Adicionalmente também foram considerados critérios presentes em estudos anteriores de grande relevância. Em função disso, esta seção tem por objetivo justamente discutir detalhadamente esses critérios, como também demonstrar a metodologia de busca utilizada para ampliar a compreensão do leitor.

3.1. Metodologia de busca

Para conduzir o processo de busca foram definidas as seguintes bases de dados de alto impacto em ciência da computação: *(i)* ACM Digital Library; *(ii)* IEEE Xplore; *(iii)* Ingenta Connect; *(iv)* Science Direct; *(v)*Springer Link e; *(vi)* Wiley. Ao final do processo de busca foram obtidos 35 trabalhos. Após a filtragem por título e relevância restaram 9 soluções que foram apropriadamente incluídas e revisadas neste estudo.

3.2. Critérios de classificação

A partir da revisão das soluções de defesa encontradas, foram identificadas três características fundamentais para prover uma categorização consistente das respectivas soluções, são elas: *(i)* Estratégia de detecção; *(ii)* Estratégia de mitigação e; *(iii)* Local de implantação da solução. Estes critérios foram considerados nos estudos anteriores: [Dayal et al. 2016], [Yan et al. 2016] e [Lohachab and Karambir 2018]. Todavia cada autor os considerou de forma individual, enquanto que o presente estudo classifica as soluções de mitigação aqui encontradas na perspectiva dos três critérios em conjunto, fornecendo uma classificação mais ampla e detalhada. Além disso, foram descritas as desvantagens encontradas em cada mecanismo de defesa, conforme é possível observar na tabela 2.

3.2.1. Local de Implantação da solução

As soluções de defesa contra ataques DDoS, podem ser classificadas com base em diversos aspectos, entre eles estão a localização da rede onde são implementadas, como é descrito em [Yan et al. 2016]. Neste sentido, foi observado que os autores optaram por realizar a implantação de duas formas distintas:

- **Na borda da rede:** Pertencem a essa categoria mecanismos locais implementados em um único componente na borda da rede com a finalidade de monitorar tanto o tráfego proveniente quanto o que chega na rede.
- **Distribuída:** Esta categoria engloba as abordagens, implementadas em vários componentes distribuídos em domínios de rede diferentes.

3.2.2. Estratégias de detecção

Outra característica levada em consideração na classificação das soluções é a estratégia utilizada durante o processo de detecção de ataques DDoS, que podem ser divididos da seguinte forma, segundo [Dayal et al. 2016]:

- **Análise estatística:** O objetivo desta técnica é analisar características do tráfego que chega ao sistema e desenvolver um modelo estatístico que define o limite de tráfego admissível. Entretanto esse modelo pode ser construído de forma estática. Em caso haja um desvio desse comportamento, o processo de mitigação é disparado.
- **Politicas pré-definidas:** Esta técnica funciona a partir da implementação de políticas de segurança previamente estabelecidas pelos administradores da rede, em grande parte estáticas. Em seguida todos os fluxos que chegam ao sistema são submetidos a essas políticas e caso não atendam seus requisitos, serão considerados como maliciosos.
- **Machine Learning:** Os algoritmos presentes nesta categoria utilizam um conjunto de dados de treinamento para se adaptar gradativamente as mudanças ocorridas no sistema e tomar decisões em tempo real a respeito da melhor ação de mitigação que deve ser aplicada com base em diversas situações de ameaças. Entre eles estão: algoritmos genéticos, lógica fuzzy, redes bayesianas, redes neurais e etc.

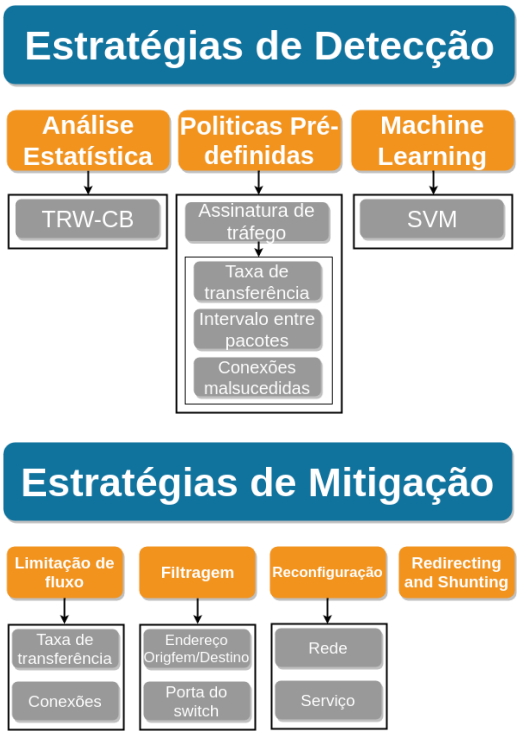


Figura 1. Visão geral das estratégias de detecção e mitigação utilizadas pelas soluções revisadas.

3.2.3. Estratégias de Mitigação

Por fim foi levado em consideração, a estratégia de mitigação empregada em cada solução como critério de classificação. Na figura 1 é apresentada de forma esquematizada um resumo principais estratégias de detecção e mitigação utilizadas. Neste sentido o trabalho desenvolvido em [Shameli-Sendi et al. 2015], também apresenta uma extensa classificação e foi utilizado como base no presente estudo, conforme mostrado abaixo:

- **Limitação de fluxo:** O objetivo desta técnica é descartar a fração de tráfego que excede a capacidade total que o sistema é capaz de lidar, mantendo a disponibilidade do mesmo.
- **Filtragem:** Nesta técnica, é realizado o bloqueio dos pacotes que entram no sistema, com base em uma ou várias características de tráfego previamente selecionadas.
- **Pushback:** Nesta estratégia, diversos domínios colaboram entre si encaminhando as informações referentes a um ataque afim de limitar a taxa de fluxos o mais próximo possível de o mais próximo possível de sua origem.
- **Reconfiguração:** Está técnica pode ser dividida em três grupos principais: *(i)* Reconfiguração de rede, onde a topologia é alterada provendo rotas alternativas para evitar o congestionamento causado pelos ataques DDoS;*(ii)* Reconfiguração de serviço, onde novas instâncias do serviço alvo são criadas em diferentes localizações.
- **Redireting and Shunting:** Nesta técnica os fluxos apontados como maliciosos, ao invés de serem imediatamente descartados são encaminhados para um dispositivo capaz de realizar uma análise mais profunda, com o intuito de extrair informações relevantes e realizar a extração do tráfego malicioso e encaminhando o tráfego benigno em direção ao seu destino final.

4. Soluções de Mitigação de DDoS baseadas em SDN

Os autores em [Bull et al. 2016] propuseram a mitigar diversos ataques DDoS, através da implementação de uma solução baseada em três componentes principais: *(i)* Switches Openflow; *(ii)* Gerenciador de estatísticas e;*(iii)* um conjunto de ações de mitigação. Os switches Openflow são um elemento primordial, atuando no encaminhamento de fluxos para o Gerenciador de estatísticas que irá coletar essas informações e fornecer recursos para que o controlador selecione a melhor ação de mitigação a ser aplicada, que pode ser uma das seguintes: *(i)* Bloquear efetivamente um fluxo ; *(ii)* Encaminhar um fluxo para quarentena para uma análise profunda, antes decidir seu devido tratamento e; *(iii)* Limitar a taxa de fluxo afim de minimizar os impactos causados a rede. Todavia os autores não realizaram uma discussão aprofundada sobre os detalhes de implementação dos mecanismos, como os métodos de classificação que oram empregados para geração dos alertas de ataque. Ademais, a efetividade do mecanismo proposto foi validade no escopo de poucos tipos de ataques DDoS.

O trabalho desenvolvido por [Sharma et al. 2017] apresenta o framework *Dist-BlockNet*, uma arquitetura de segurança distribuída que emprega as tecnologias blockchain e SDN para proteger ambientes IoT. Os autores desenvolveram uma arquitetura modular capaz de prover escalabilidade e flexibilidade excluindo a necessidade de um controlador central, que em muitos casos pode implementar um ponto único de falha. Além disso o framework é capaz de se adaptar automaticamente a diversos cenários de ameaças, como ataques DDoS, envenenamento ARP(ARP Poisoning) e ataques de falsificação de topologia (Fake Topology Attacks). A arquitetura é baseada em dois módulos principais: *(i)* Orchapp e; *(ii)* Shelter. O Orchapp é responsável por implementar dinamicamente as politicas de segurança aplicadas ao sistema. Enquanto o módulo Shelter atua principalmente na camada de dados, coletando informações de tráfego e gerando diagramas com estatísticas de fluxo que são fornecidos ao Orchapp. Após colher essas estatísticas o Or-

chapp irá utilizar um dos mecanismos de detecção disponíveis para verificar a ocorrência de um ataque, e em caso positivo atualizar por meio da técnica de blockchain todas as tabelas de fluxo do sistema com as medidas de segurança mais adequadas. Apesar de ser capaz de selecionar adequadamente politicas de segurança em diferentes cenários de ameaças, o framework depende que as politicas de segurança sejam especificadas pelo administrador da rede, caso contrário não irá funcionar corretamente na contenção de ameaças

[Bhunia and Gurusamy 2017] desenvolveu um framework hierárquico denominado *SoftThings* que assim como o mecanismo revisado anteriormente se propõe a mitigar uma ampla gama de ataques DDoS. Todavia, focando em redes de dispositivos IoT domiciliares. A arquitetura do *SoftThings* é segmentada em quatro subníveis: *(i)* Dispositivos IoT; *(ii)* Switches SDN; *(iii)* Controlador de cluster SDN e; *(iiii)* Controlador SDN mestre. A base da hierarquia é formada por clusters de dispositivos IoT interconectados ao mesmo switch SDN. Para cada cluster é atribuído um Controlador de Cluster SDN responsável pelo seu gerenciamento. Por sua vez cada Controlador de Cluster SDN tem de informar periodicamente o estado de sua rede para o Controlador SDN Mestre que está no topo da hierarquia do *SoftThings* e é responsável por tomadas de decisões globais que serão aplicadas a toda infraestrutura.

O processo de detecção é iniciado quando o Controlador de cluster SDN, reconhece um comportamento anômalo, com base nas estatísticas de tráfego oferecidas pelos switches que monitoram o tráfego dos dispositivo IoT conectado a ele. No Controlador de cluster SDN residem três módulos: *Learning module*, *Classification module* e *Flow Management Module*. O *Learning Module* utiliza as estatísticas de tráfego em condições normais coletadas dos switches OpenFlow e padrões de tráfego de ataque, para continuamente alimentar o *Classification module* com dados de treinamento. O *Classification module* utiliza o algoritmo Support Vector Machine (SVM), para detecção de anomalias e informa ao *Flow management module* que seleciona a medida de segurança apropriada, que pode ser: *(i)* Envio do endereço malicioso para ser adicionado em uma blacklist no Controlador SDN Mestre e; *(ii)* bloqueio total do fluxo malicioso nos switches OpenFlow. Apesar do framework ser focado em garantir escalabilidade, através da existência de diversos controladores para gerenciar clusters de dispositivos SDN o Controlador SDN Mestre representa um ponto único de falha e pode colocar em risco a disponibilidade de toda a infraestrutura.

Em contraste com as abordagens citadas anteriormente, que implementam proteção contra ameaças internas e externas, [Ozcelik et al. 2017] propôs uma solução que se destina unicamente a monitorar o tráfego proveniente de redes IoT, afim de detectar comportamentos anômalos. Segundo os autores quanto mais próximo da fonte, mais eficiente será o processo de detecção e mitigação. O mecanismo implementa pequenas infraestruturas na borda de redes alvos tem a capacidade de analisar todo o tráfego de saída e determinar sua legitimidade. Em caso de fluxos maliciosos, o mecanismos realiza o bloqueio total do tráfego ilegítimo. Para tal fim é utilizado o algoritmo TRW-CB (Threshold Random Walk with Credit Based Rate Limiting) que verificar a legitimidade dos fluxos analisados. Porém o processo de mitigação é desencadeado com base em uma quantidade estática de conexões malsucedidas por segundo. É importante ressaltar que a utilização dessa abordagem pode implicar em um processo de detecção ineficiente, em casos onde o

atacante emite uma pequena quantidade de tráfego, imitando o comportamento de clientes autêntico.

Em [Yin et al. 2018] os autores desenvolveram o framework SD-IoT, e desenvolveram um método de detecção baseado no algoritmo *cosine similarity*. O ponto chave da implementação do algoritmo descrito pelos autores é comparar amostras de pacotes do tipo *Packet-ins* que chegam periodicamente ao controlador com um limite ideal pré-definido. Os autores denominam essas amostras como vetores. A diferença entre os vetores deve ficar abaixo do límite pré-determinado, intitulado de semelhança de cosenos, que representa um valor entre 0 e 1. Caso essa condição não seja alcançada, isso pode indicar que um ataque DDoS está em curso na rede. Em seguida é gerado um alarme para que o processo de mitigação seja disparado. Após a identificação dos fluxos maliciosos o mecanismo identifica qual a porta e switch Openflow de origem dos fluxos e aplica a política de bloqueio. Apesar da eficácia da abordagem proposta, os autores deixam claro que o processo de determinação do valor limite de semelhança é complexo e deve ser feito adequadamente. Pois se esse valor for demasiado alto, parte do tráfego DDoS será classificado como benigno, enquanto que se for muito baixo, pacotes normais serão classificados como maliciosos, comprometendo assim a eficácia do processo de mitigação.

Outra abordagem que empregou gateways SDN para proteger infraestruturas IoT, foi desenvolvida em [Yan et al. 2018] onde foi proposto um framework distribuído denominado MLDMF (multi-level DDoS mitigation framework). Os autores se concentraram em desenvolver uma abordagem escalável e tolerante a falhas. Em função disso o framework é dividido em três camadas principais: *(i)* Computação de borda; *(ii)* Computação em névoa e; *(iii)* Computação em nuvem. A camada de computação de borda está diretamente conectada aos dispositivos IoT é implementada por um gateway SDN responsável principalmente por garantir o controle de acesso, a comunicação de dados segura e a aplicação das medidas de segurança provenientes da camada de computação em névoa, que está em um nível logo acima e tem como tarefa primordial coletar as estatísticas de tráfego e enviá-los para a camada de nuvem no topo da infraestrutura, onde são definidas as politicas de segurança a nível de aplicação que serão aplicadas globalmente. Apesar de se propor a prevenir proativamente diversas ameaças, quando se trata de ataques externos, o mecanismo foi avaliado em cenários de ataques de pequena escala. Portanto, é sugerido também a implementação do mecanismo em cenários críticos, com grandes quantidades de tráfego afim de garantir sua eficácia inclusive nesssas situações.

Seguindo a perspectiva dos dois trabalhos anteriores em [Krishnan et al. 2018] foi implementada uma arquitetura para gerenciamento de redes IoT em larga escala. Provendo segurança e escalabilidade por meio da integração de gateways e controladores SDN na borda de redes IoT. O gerenciamento de alto nível é realizado a partir da camada de nuvem que atua na definição de politicas de segurança e as transmite à camada de névoa para que esta possa aplicá-la aos dispositivos IoT gerenciados na base da infraestrutura. Os autores realizaram um estudo sobre sua utilização na defesa contra ataques DDoS do tipo HTTP Flooding. Para este fim foram utilizados um switch SDN conectado a um dispositivo executando uma aplicação HTTP. Foi instalada uma regra de fluxo no switch, que disparava um alerta de ataque, caso o número de tentativas de conexão ao servidor excedesse o limite de 350 tentativas por segundo. Após a comprovação do ataque todas conexões ilegítimas são descartadas e uma mensagem de redirecionamento de endereço

do servidor é enviada para todos os clientes. Caso algum cliente mal-intencionado receba a mensagem, espera-se que ele não seja capaz de decodificá-la, continuando o ataque ao endereço original. Apesar de desenvolverem uma abordagem capaz de prover segurança em ambientes IoT de larga escala, a utilização de mecanismos pouco flexíveis, dificulta a aplicabilidade da solução em ambientes heterogêneos.

No trabalho desenvolvido em [Sharma et al. 2018] foi proposta uma arquitetura genérica modular denominada *ShSec*, que atua como um middleware, provendo segurança em casas inteligentes. A arquitetura oferece prevenção a incidentes internos e externos, atuando na integração entre as camadas de aplicação e dados SDN. O mecanismo é composto por dois componentes principais: *(i)* Orchestrator e; *(ii)* KNOT. O módulo Orchestrator é responsável por manter interoperabilidade entre a camada de aplicação SDN e a arquitetura ShSec, provendo comunicação de dados confiável. Enquanto que o módulo KNOT implementa as funções de proteção contra ameaças de rede. Sempre que um novo fluxo chega ao controlador na camada de aplicação, o gráfico de fluxo da rede é atualizado e é realizada uma checagem ao banco de dados de ataques, para constatar se os fluxos correspondem a um ataque conhecido. Caso positivo o controlador irá gerar um alarme e instalar a política de bloqueio adequada, para realizar a mitigação dos fluxos maliciosos. Caso negativo o fluxo é enviado para uma inspeção mais profunda, afim de atualizar o banco de dados de ataque conhecidos. Apesar de se propor a prevenir proativamente diversas ameaças, quando se trata de ataques externos, o mecanismo foi avaliado em cenários de ataques de pequena escala. Portanto, assim como no trabalho desenvolvido em [Yan et al. 2018] também é necessária a implementação do presente framework em cenários de larga escala, afim de avaliar sua eficácia nessas situações.

Em [Salva-Garcia et al. 2018] e [Molina Zarca et al. 2018] foi apresentado um framework distribuído de segurança utilizando mecanismos filtragem de tráfego para fornecer funcionalidades de controle de acesso e gerenciamento em redes de dispositivos IoT, fazendo uso da integração das tecnologias SDN e NFV. A abordagem implementa politicas de filtragem, que são proativamente especificadas pelo administrador da rede utilizando uma interface gráfica de suporte localizada na camada de administração no topo da arquitetura. Em seguida essas politicas são traduzidas em configurações de baixo nível e o plano de orquestração de segurança se encarrega de encaminhá-las ao plano de execução de segurança, aqui implementado por diversos gateways SDN, residindo na borda da rede de destino. Em cada um desses gateways SDN residem agentes de filtragem e monitoramento responsáveis pelo processo de detecção e mitigação de ameaças. O agente de monitoramento realiza a confirmação da existência de um ataque DDoS utilizando como base um banco de dados de assinaturas de ataques pré-configuradas. Em seguida, caso os fluxos coincidam com um dos padrões é disparado um alarme para o agente de filtragem, que irá reagir bloqueando todos os fluxo maliciosos. Apesar da arquitetura ser capaz de prover segurança em ambientes IoT, a necessidade manual de atualizações das politicas de filtragem pelo administrador da rede, torna necessário a especificação de mecanismos mais flexíveis e auto gerenciáveis.

5. Classificação das Soluções

A tabela 2 apresenta as soluções de mitigação descritas na seção anterior, classificadas com base na taxonomia proposta. Comumente, ataques DDoS geram uma quantidade significativa de tráfego na rede. Portanto, uma ideia natural para detectar tais ataques é

identificar variações abruptas no volume de tráfego da rede. Nesse sentido, pode-se observar que a maioria das soluções (70%) adota métodos estatísticos para modelar o fluxo de tráfego normal da rede. Se o comportamento observado desviar significativamente desse modelo, então suspeita-se que o sistema está sob ataque. Alguns autores aplicam um limiar estático para distinguir uma situação de ataque de uma situação real, enquanto que outros adotam um limiar dinâmico para refletir as variações do padrão de tráfego que podem ocorrer em diferentes períodos do dia. Uma limitação do uso de limiar estático é que a defesa precisa passar por uma etapa de treinamento antes de realmente ser testada em uma situação real. Um limiar dinâmico pode ser enganado por atacantes inteligentes que expandem gradualmente o tráfego do ataque, aumentando o limiar para acompanhar o aparente crescimento do tráfego legítimo.

Solução	Detecção	Mitigação	Localização	Limitações
[Bull et al. 2016]	Análise estatística	Filtragem, Redirecting and Shunting, Limitação de fluxo.	Borda da rede	Os autores não especificam com profundidade os detalhes de implementação do mecanismo de detecção; Validação com poucos ataques.
[Sharma et al. 2017]	Políticas pré-definidas	Filtragem	Distribuída	Os autores não especificam com profundidade os detalhes de implementação do mecanismo de detecção e mitigação.
[Bhunia and Gurusamy 2017]	Machine Learning	Filtragem e Limitação de Fluxo	Distribuída	O Master SDN Controller implementa um ponto único de falha, impactando na disponibilidade de toda a infraestrutura
[Ozcelik et al. 2017]	Análise estatística	Filtragem	Distribuída	Detecção baseado em parâmetros estáticos.
[Yin et al. 2018]	Análise estatística	Filtragem	Distribuída	O cálculo do valor ideal de similaridade de cosenos é complexo e impacta negativamente na eficiência do mecanismo, caso não seja feita adequadamente.
[Yan et al. 2018]	Análise estatística	Filtragem	Distribuída	Mecanismo não avaliado em cenários de ataques de larga escala.
[Krishnan et al. 2018]	Análise estatística	Reconfiguração	Distribuída	Detecção baseada em parâmetros estáticos.
[Sharma et al. 2018]	Análise estatística	Filtragem, Redirecting and Shunting	Borda da rede	Mecanismo não avaliado em cenários de ataques de larga escala.
[Salva-Garcia et al. 2018]	Políticas pré-definidas	Filtragem	Distribuída	Necessidade de atualização manual das politicas de filtragem.

Tabela 2. Sumário das soluções de mitigação de ataques DDoS usando SDN, aplicadas a ambientes IoT.

No que diz respeito as ações mitigadoras de risco, a filtragem de tráfego é empregada com uma maior frequência. Embora essa defesa possa ser eficaz na mitigação de casos específicos de um ataque, dentre suas desvantagens destaca-se o fato de que muitas soluções assumem que um endereço IP esta associado a um único cliente. No entanto, tal hipótese não pode ser mantida quando vários clientes são multiplexados por trás de um NAT ou proxy. Um ataque realizado por poucos clientes mal-intencionados pode resul-

tar no bloqueio do serviço para todas as solicitação por trás do endereço IP do roteador NAT ou proxy, impedindo que outros usuários honestos usem o serviço. Por fim, foi observado que cerca de 80% das soluções implementam frameworks distribuídos em várias camadas para prover maior escalabilidade e flexibilidade. Entretanto cada camada pode se tornar um alvo em potencial dos atacantes, influenciando na disponibilidade de toda infraestrutura.

Portanto, é possível concluir que a detecção e mitigação de ataques DDoS baseados em IoT continua sendo um tema de pesquisa relevante e desafiador e que a tecnologia SDN deve ser melhor explorada para prover uma soluções mais robusta contra esses ataques.

6. Conclusão

Neste trabalho foi proposta uma taxonomia das principais soluções existentes de defesa contra ataques DDoS usando o paradigma SDN em ambientes IoT. Isso foi viabilizado mediante a definição de critérios capazes de auxiliar na compreensão das contribuições oferecidas pelas abordagens revisadas. Adicionalmente foi apresentada uma classificação satisfatória das estratégias de detecção e mitigação empregadas em cada solução. Por fim, com base nos resultados obtidos foi comprovada a eficacia do paradigma SDN, como um meio de proteger ambientes IoT contra ataques DDoS.

Todavia ainda existem questionamentos em aberto como: *(i)* A própria segurança da infraestrutura SDN, diante de ameaças de alto risco, como os ataques DDoS e; *(ii)* Aplicabilidade em cenários reais das soluções de mitigação desenvolvidas. Estes questionamentos serão abordados em uma taxonomia futura que irá abranger um maior número de critérios de classificação e soluções.

7. Agradecimentos

Esta pesquisa foi parcialmente apoiada pela 4ª chamada colaborativa EU-BR (H2020, *grant agreement* no. 777067, NECOS – *Novel Enablers for Cloud Slicing*), financiada pela Comissão Européia e pelo Ministério da Ciência, Tecnologia, Inovação e Comunicação (MCTIC), através da RNP e CTIC. Os autores também agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e ao IFRN pelo apoio.

Referências

[Al-Fuqaha et al. 2015] Al-Fuqaha, A. I., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys and Tutorials*, 17(4):2347–2376.

[Bertino and Islam 2017] Bertino, E. and Islam, N. (2017). Botnets and internet of things security. *Computer*, 50(2):76–79.

[Bhunia and Gurusamy 2017] Bhunia, S. S. and Gurusamy, M. (2017). Dynamic attack detection and mitigation in iot using sdn. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6.

[Bull et al. 2016] Bull, P., Austin, R., Popov, E., Sharma, M., and Watson, R. (2016). Flow based security for iot devices using an sdn gateway. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 157–163.

[Ceron et al. 2016] Ceron, J. M., Margi, C. B., and Granville, L. Z. (2016). Mars: An sdn-based malware analysis solution. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 525–530.

[Cherian and Chatterjee 2019] Cherian, M. and Chatterjee, M. (2019). Survey of security threats in iot and emerging countermeasures. In Thampi, S. M., Madria, S., Wang, G., Rawat, D. B., and Alcaraz Calero, J. M., editors, *Security in Computing and Communications*, pages 591–604, Singapore. Springer Singapore.

[Dayal et al. 2016] Dayal, N., Maity, P., Srivastava, S., and Khondoker, R. (2016). Research trends in security and ddos in sdn. *Security and Communication Networks*, 9(18):6386–6411.

[Farris et al. 2019] Farris, I., Taleb, T., Khettab, Y., and Song, J. (2019). A survey on emerging sdn and nfv security mechanisms for iot systems. *IEEE Communications Surveys Tutorials*, 21(1):812–837.

[Kalkan et al. 2017] Kalkan, K., Gur, G., and Alagoz, F. (2017). Defense mechanisms against ddos attacks in sdn environment. *IEEE Communications Magazine*, 55(9):175–179.

[Kalkan and Zeadally 2018] Kalkan, K. and Zeadally, S. (2018). Securing internet of things with software defined networking. *IEEE Communications Magazine*, 56(9):186–192.

[Kanagavelu and Aung 2019] Kanagavelu, R. and Aung, K. M. M. (2019). A survey on sdn based security in internet of things. In Arai, K., Kapoor, S., and Bhatia, R., editors, *Advances in Information and Communication Networks*, pages 563–577, Cham. Springer International Publishing.

[Kolias et al. 2017] Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J. (2017). Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84.

[Koufopavlou 2015] Koufopavlou, E. H. K. P. S. D. J. H. S. D. M. O. (2015). Software-Defined Networking (SDN): Layers and Architecture Terminology. RFC 7426.

[Kouicem et al. 2018] Kouicem, D. E., Bouabdallah, A., and Lakhlef, H. (2018). Internet of things security: A top-down survey. *Computer Networks*, 141:199 – 221.

[Krishnan et al. 2018] Krishnan, P., Najeem, J. S., and Achuthan, K. (2018). Sdn framework for securing iot networks. In Kumar, N. and Thakre, A., editors, *Ubiquitous Communications and Network Computing*, pages 116–129, Cham. Springer International Publishing.

[Lohachab and Karambir 2018] Lohachab, A. and Karambir, B. (2018). Critical analysis of ddos—an emerging security threat over iot networks. *Journal of Communications and Information Networks*, 3(3):57–78.

[Marzano et al. 2018] Marzano, A., Alexander, D., Fonseca, O., Fazzion, E., Hoepers, C., Steding-Jessen, K., Chaves, M. H. P. C., Cunha, , Guedes, D., and Meira, W. (2018). The evolution of bashlite and mirai iot botnets. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00813–00818.

[Molina Zarca et al. 2018] Molina Zarca, A., Bernal Bernabe, J., Farris, I., Khettab, Y., Taleb, T., and Skarmeta, A. (2018). Enhancing iot security through network softwariza-

tion and virtual security appliances. *International Journal of Network Management*, 28(5):e2038. e2038 nem.2038.

[Noor and Hassan 2019] Noor, M. and Hassan, H. (2019). Current research on internet of things (iot) security: A survey. *Computer Networks*, 148:283–294.

[Ozcelik et al. 2017] Ozcelik, M., Chalabianloo, N., and Gür, G. (2017). Software-defined edge defense against iot-based ddos. In *2017 IEEE International Conference on Computer and Information Technology (CIT)*, pages 308–313.

[Salva-Garcia et al. 2018] Salva-Garcia, P., Alcaraz-Calero, J. M., Wang, A., Qi, B., Bernal, J., and Skarmeta, A. (2018). 5g nb-iot: Efficient network traffic filtering for multitenant iot cellular networks. *Security and Communication Networks*.

[Shameli-Sendi et al. 2015] Shameli-Sendi, A., Pourzandi, M., Fekih-Ahmed, M., and Cheriet, M. (2015). Taxonomy of distributed denial of service mitigation approaches for cloud computing. *Journal of Network and Computer Applications*, 58:165 – 179.

[Sharma et al. 2018] Sharma, P. K., Park, J. H., Jeong, Y.-S., and Park, J. H. (2018). Shsec: Sdn based secure smart home network architecture for internet of things. *Mobile Networks and Applications*.

[Sharma et al. 2017] Sharma, P. K., Singh, S., Jeong, Y., and Park, J. H. (2017). Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks. *IEEE Communications Magazine*, 55(9):78–85.

[Shi et al. 2017] Shi, Y., Dai, F., and Ye, Z. (2017). An enhanced security framework of software defined network based on attribute-based encryption. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, pages 965–969.

[Yakasai and Guy 2015] Yakasai, S. T. and Guy, C. G. (2015). Flowidentity: Software-defined network access control. In *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pages 115–120.

[Yan et al. 2018] Yan, Q., Huang, W., Luo, X., Gong, Q., and Yu, F. R. (2018). A multi-level ddos mitigation framework for the industrial internet of things. *IEEE Communications Magazine*, 56(2):30–36.

[Yan et al. 2016] Yan, Q., Yu, F. R., Gong, Q., and Li, J. (2016). Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys Tutorials*, 18(1):602–622.

[Yin et al. 2018] Yin, D., Zhang, L., and Yang, K. (2018). A ddos attack detection and mitigation with software-defined internet of things framework. *IEEE Access*, 6:24694–24705.

[Zargar et al. 2013] Zargar, S. T., Joshi, J., and Tipper, D. (2013). A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE Communications Surveys Tutorials*, 15(4):2046–2069.

Realização

Organização

Apoio Institucional

Patrocínio

Diamante

Ouro

Prata

Bronze