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.

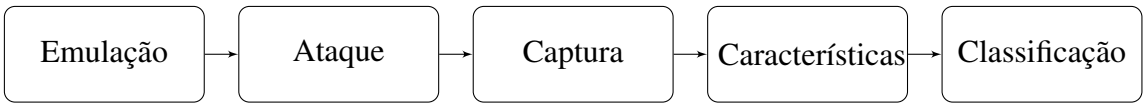

Figura 1. Fluxo dos experimentos

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.

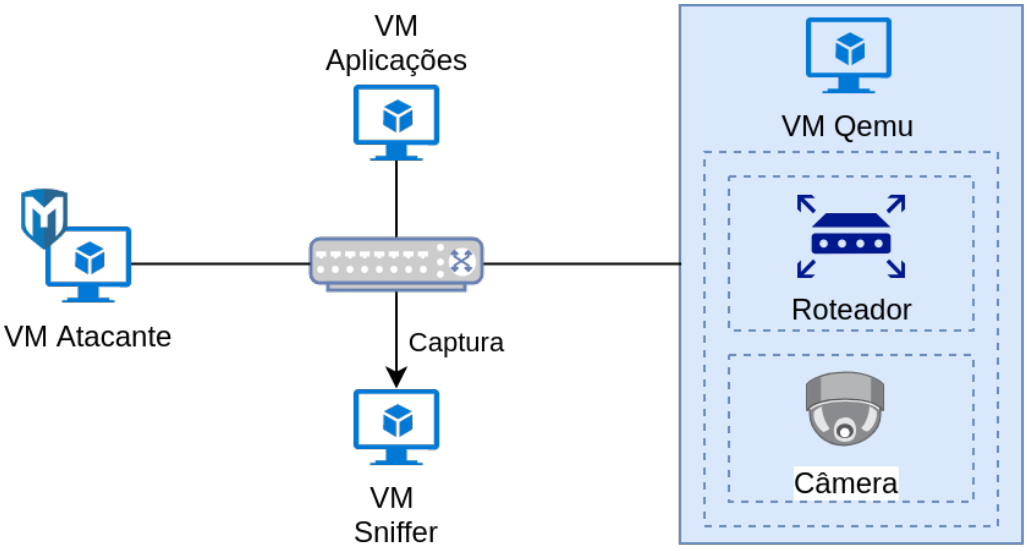

Figura 2. Topologia utilizada para a realização dos experimentos

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].

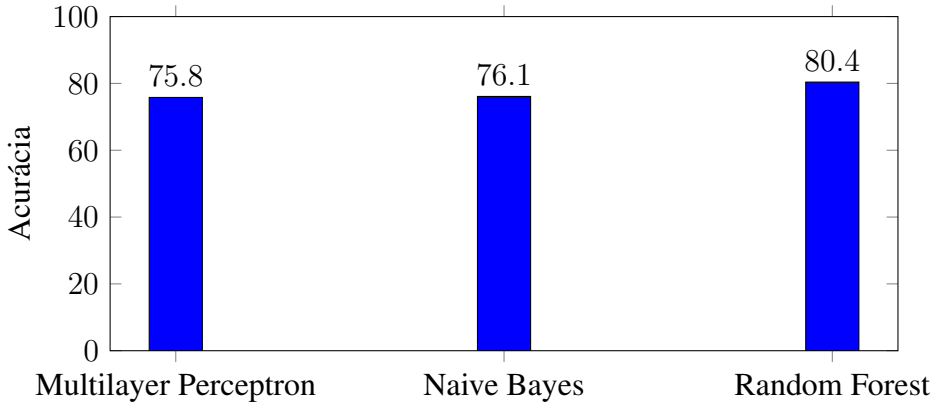

Figura 4. Performance dos algoritmos aplicados na coleção KDD

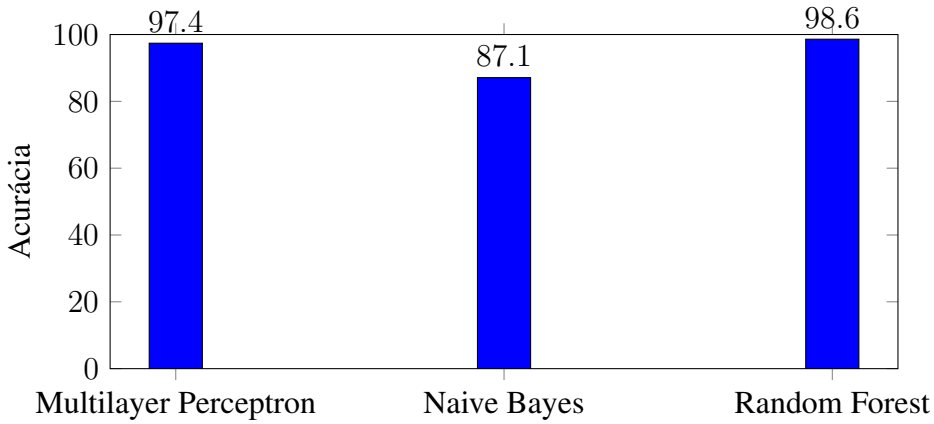

Figura 5. Performance dos algoritmos aplicados na nova coleção

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.