UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA IMPLEMENTAÇÃO PARALELA DO ALGORITMO DE DETECÇÃO DE NOVIDADE EM STREAMS MINAS

LUÍS PUHL

ORIENTADOR: PROF. DR. HERMES SENGER

São Carlos – SP

Fevereiro/2020

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA IMPLEMENTAÇÃO PARALELA DO ALGORITMO DE DETECÇÃO DE NOVIDADE EM STREAMS MINAS

LUÍS PUHL

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Sistemas de Computação
Orientador: Prof. Dr. Hermes Senger

São Carlos – SP

Fevereiro/2020

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Paralelize o minas no flink. (não se preocupe com o uso, seja ele NIDS ou qualquer outra coisa) Use a detecção de intrusão apenas como validação do algoritmo.

cap 1: - Objetivo paralelizar minas em plataforma de big-data capaz de consumir streams de forma eficiente - Motivação: Minas é recente, com potencial em várias aplicações, por exemplo (NIDS, sensores, ...) para isso deseja-se uma implementação eficiente (low power, ou usar todo hardware) e escalável (big-data)

A Internet das Coisas (*Internet of Things* - IoT) é um tema frequentemente abordado em pesquisas de segurança em redes de computadores claramente motivado pelo crescimento dessa rede, estimada em 8.4 bilhões de dispositivos em 2017 [1] e, pelo risco que ela representa fundamentado no histórico de ataques massivos realizados por milhares de nós subvertidos como o realizado pela *botnet* mirai em 2016 [2]. Mais preocupante nesse cenário são os fatores que possibilitaram esses ataques: falta de controle sobre a origem do hardware e software embarcado nos dispositivos além das cruciais atualizações de segurança.

Com esse desafio de segurança, especialmente em IoT industrial onde a subversão de dispositivos pode causar danos reais imediatamente, profissionais de segurança de redes e operadores de redes são confrontados com enorme superfície de ataque composta por diversos sistemas, tecnologias com longo tempo de vida e poucas atualizações, um sistema de detecção de intrusão (*Intrusion DetectionSystem* - IDS) operando na rede de computadores local torna-se uma ferramenta muito importante para defesa dessa rede e os serviços que ela suporta.

Os IDS foram tradicionalmente construídos à partir de técnicas de mineração de dados (*Data Mining* - DM), aprendizado de máquina (*Machine Learning* - ML), mais especificamente Detecção de Novidades (*Novelty Detection* - ND) para detectar ataques e descobrir novos padrões, porém ao analisar tráfego de rede a velocidade da análise deve ser próxima ou superior

à velocidade da rede analisada além de não consumir mais recursos do que a própria rede analisada. Mais restrições nesse sentido devem ser incorporadas quando trata-se de uma rede IoT, diferente de uma rede operando em computação na nuvem (*Cloud Computing* - cloud), especialmente latência e banda são ainda mais restritos e ao mitigar esses atributos movendo o IDS para o mais próximo da rede IoT passando a processar na névoa computacional (*Fog Computing* - fog) armazenamento e processamento são também restringidos. Portanto uma única leitura do conjunto analisado, rápida atualização e distribuição do modelo de detecção e resultados em tempo real são características positivas encontradas nas técnicas de mineração e processamento de fluxos de dados (*Data Streams* - DS).

Nesse contexto, foca-se na arquitetura de IDS proposta por [3] baseada no algoritmo de ND em DS MINAS [4] e na implementação BigFlow [5] proposta para redes 10 *Gigabit Ethernet*. Acredita-se que a fusão dessas abordagens em uma nova implementação seja capaz de tratar uma rede de maior fluxo com nível comparável de precisão da análise com o mesmo hardware e maior capacidade de escalonamento horizontal com distribuição de carga entre nós na Fog.

Ideias:

Necessidade de utilizar recursos da Fog para eliminar latência de comunicacao com a nuvem (enviar os dados para classificar na nuvem teria alta latencia)

Necessidade de processamento concorrente e distribuído para ter maior escalabilidade

1.1 Objetivos

Avaliar o algoritmo minas na plataforma Flink para detecção de intrusão em redes IoT.

Objetivos secundários: Identificar métricas na literatura: - Quanto a detecção de anomalias; - e desempenho na detecção de intrusão da literatura; Implementar Minas sobre Apache Flink para detecção de intrusão em redes IoT; Extrair as métricas de detecção de anomalias da implementação com datasets diferentes; Validar a implementação comparando as métricas extraídas da implementação com as encontradas na literatura; Extrair as métricas de desempenho da implementação com datasets diferentes;

1.2 Motivação

- flink não foi abordado para intrusões em redes iot; - contribuição de técnica/método para a area de segurança da informação;

Implementar a arquitetura proposta em [3] (IDS-IoT) com foco no volume de fluxo explorando aspectos de paralelismo e distribuição de recursos em fog e cloud visando alta vazão e baixa latência.

Para redução da latência pretende-se utilizar recursos próximos à rede monitorada (fog) distribuindo o volume entre vários nós (com recursos muitas vezes limitados) para dar vazão ao processamento. Além disso, para tarefas com maiores exigências de recursos de processamento e armazenamento como treinamento e reconstrução do modelo pretende-se utilizar computação em nuvem.

Implementar o algoritmo MINAS [4] buscando máximo desempenho nos aspectos latência e vazão utilizando recursos de fog e nuvem além de avaliar seus resultados qualitativos na detecção e classificação de novidades em *datasets* relevantes a detecção de intrusão em redes IoT.

1.3 Proposta Metodológica

Implementar – e avaliar esta implementação – o algoritmo MINAS [4] aplicando técnicas, como Apache Flink, usadas na solução BigFlow [5]; Aplicar essa implementação na arquitetura proposta por [3] e comparar o desempenho com a implementação e conjunto de dados (dataset) original; Avaliar o desempenho e escalabilidade da aplicação com dataset MAWIFlow.

Flink - dizer que vai usar, para fazer o que e porque escolheu

Kafka ?

Raspberry - para todos

A base -

Capítulo 2

FUNDAMENTOS CIENTÍFICOS E TECNOLÓGICOS

2.1 Computação em Nuvem, Borda e Névoa

Definir internet e rede (borda, centro, etc) Classificando a internet por sua densidade podemos dizer que ao centro estão os *data centers* e nuvems públicas em seguida o núcleo interconectando redes diversas, redes locais e a Borda composta pelos nós folha dentro de uma rede local.

O modelo de Computação em Nuvem (*Cloud Computing*) permite alocar recursos como redes, servidores, armazenamento, aplicações e serviços de maneira conveniente e seu provisionamento ágil concede elasticidade para atender demandas variáveis com custo mínimo (MELL; GRANCE, 2012).

Alternativamente, a Computação na Borda (*Edge Computing*) destaca-se no processamento em tempo real de dados originários da própria borda além de atender preocupações de segurança e privacidade (SHI et al., 2016).

Edge é por vezes chamado de Computação em Névoa (*Fog Computing*) contudo (IEEE Communications Society, 2018) indica diferenças como exclusão do modelo *cloud* e limitação a poucas camadas (por exemplo, somente os nós folha de uma rede) no modelo *edge* em direto contraste com a inlcusão de *cloud* e hierarquias maiores. *Fog* também atende gerenciamento da rede, armazenamento e controle.

Esse modelo de computação distribuida desde os nós folha até o centro é motivado pela mudança do *statu quo* do fluxo dos dados na internet: tradcionalmente os dados são produzidos pelos dispositivos de borda imediatamente emviados à *cloud* (produção, *upstream*), que armazena e processa recursos derivados servido-os através de requisição-resposta (consumo, *downstream*) a mais clientes. Com a ampliação da Internet das Coisas (*Internet of Things*, IoT)

e consequente ampliação sem precedentes do volume de dados gerados, mudando o relação de consumo e produção (SHI et al., 2016), arquiteturas tradicionais como *cloud* podem não ser capazes de lidar com esses dados por falta de banda o que leva as propostas de distribuição vertical do processamento em *fog* (Dastjerdi; Buyya, 2016; BONOMI et al., 2012).

2.2 Mineração de Dados

2.3 Mineração de Stream

- quem são, o que consomem (BigFlow apud Gaber2005) Mining data streams: A Review.

2.4 Plataformas de processamento distribuído

- arq labmda, kappa, (vide guilherme) - MapReduce, Haddop, Spark, Storm

O que são e para que servem essas ferramentas

Breve descricao do MapReduce e Hadoop

Breve descricao do Spark

Apache Spark is an open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since.

Apache Spark é um framework de código fonte aberto para computação distribuída.[1] Foi desenvolvido no AMPLab da Universidade da Califórnia[2] e posteriormente repassado para a Apache Software Foundation[3] que o mantém desde então. Spark provê uma interface para programação de clusters com paralelismo e tolerância a falhas.

Apache Spark is a fast and general-purpose cluster computing system. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.

———————

Apache Spark citeZaharia é um *framework* para construção de sistemas de computação distribuída em *cluster* com garantias de tolerância a falhas (execução em computadores não confiáveis) utilizando como premissas: paralelização e localidade de dados, como

api em Python (dataframe de pandas)

2.5 Apache Flink

Breve descrição do Flink (como esse vai ser usado, precisa explicar um pouco melhor - 2 paginas pelo menos):
- arquitetura
- modelo de programacao
- 1 pequeno exemplo de codigo explicando

2.6 Detecção de Novidade

Novelty Detection

breve descricao do que sao algoritmos para DN

ver se tem algum survey e citar

2.7 O algoritmo MINAS

breve descricao do MINAS (FARIA; CARVALHO; GAMA, 2016)

ver paper da profa. Elaine

Detecção de intrusão em redes - riscos de segurança - técnicas de intrusão e tipos de ataques - mecanismo de detecção (análise de fluxo de rede -¿ detecção de anomalia) Detecção de novidades - técnicas de Detecção de novidades - MINAS (incluir métricas) - BigFlow (incluir métricas) Processamento de Streams (big data) - cloud? - redes como stream - Atraso - Kafka/Spark/Flink Redes IoT - Restrição hardware (Energia, CPU, Mem, Rede) - Consideração FOG vs Cloud

Capítulo 3

TRABALHOS RELACIONADOS

cap 3: Trabalhos relacionados - Artigos sobre o Minas - outros que paralelizaram algoritmos de mineração de dados/streams alguns online (5-10 refs) - implementação paralelas/distribuídas em dispositivos pequenos

Aqueles que contenham: - detecção de anomalia em streams - detecção de intrusão em rede com processamento de streams - BigFlow

3.1 BigFlow

Table 1 Network-level feature set used in the experiments throughout this work [18]. Types: Host-based (Host to All), Flow-based (Source to Destination, Destination to Source, Both)

Features:
- Number of Packets,
- Number of Bytes,
- Average Packet Size,
- Percentage of Packets (PSH Flag),
- Percentage of Packets (SYN and FIN Flags),
- Percentage of Packets (FIN Flag),
- Percentage of Packets (SYN Flag),
- Percentage of Packets (ACK Flag),
- Percentage of Packets (RST Flag),
- Percentage of Packets (ICMP Redirect Flag),
- Percentage of Packets (ICMP Redirect Flag),
- Percentage of Packets (ICMP Time Exceeded Flag),

- Percentage of Packets (ICMP Unreachable Flag),
- Percentage of Packets (ICMP Other Types Flag),
- Throughput in Bytes,
- Protocol

BigFlow destaca em sua secção 2 (backgroud) o processamento de streams [18, 19], a preferencia de NIDS por anomalia em contraste aos NIDS por assinatura [30, 31, 32], a variabilidade e evolução dos padrões de tráfego em redes de propósito geral [9, 11, 20], a necessidade de atualização regular do modelo classificador [8, 9, 10, 20] e o tratamento de eventos onde a confiança resultante da classificação é baixa [9, 12, 13].

Também destaca em sua secção 3 (MAWIFlow) que datasets adequados para NIDS são poucos devido o conjunto de qualidades que os mesmos devem atender como realismo, validade, etiquetamento, grande variabilidade e reprodutividade (disponibilidade pública) [8, 9, 10, 17, 38].

Para avaliar o desempenho de NIDS o dataset MAWIFlow é proposto. Originário do 'Packet traces from WIDE backbone, samplepoint-F' composto por seções de captura de pacotes diárias de 15 minutos de um link de 1Gbps entre Japão e EUA, com inicio em 2006 continuamente até hoje, anonimizados [22], etiquetados por MAWILab [8]. Desse dataset original apenas os eventos de 2016 são utilizados e desses 158 atributos são extraídas resultando em 7.9 TB de captura de pacotes. Além disso, os dados são estratificados [24] para redução de seu tamanho a um centésimo mantendo as proporções de etiquetas (Ataque e Normal) facilitando o compartilhamento e avaliação de NIDS além de atender as qualidades anteriormente mencionadas.

Com o dataset MAWIFlow original e reduzido foram avaliados quatro classificadores [42, 43, 44, 45] da literatura em dois modos de operação quanto seus dados de treinamento (ambos contendo uma semana de captura) o primeiro usando somente a primeira semana do ano e as demais como teste e o segundo modo usando a semana anterior como treinamento e a seguinte como teste. Demostrando, com 62 atributos, que a qualidade da classificação retrai com o tempo quando não há atualização frequente do modelo classificador.

Capítulo 4

IMPLEMENTAÇÃO E TESTES

4.1 Descrição da Implementação

- offline, online, ND, Clustering

- observação/Considerações de paralelização

Notas sobre implementação Python/Kafka/Minas (não escala como esperado)

Dificuldade no processamento distribuido em Flink.

- complexidade bigO (?)

4.2 Cenário de Teste

- detecção de intrusão - Arquitetura guilherme (dispositivos pequenos vs cloud) (CASSALES et al., 2019)

- BigFlow com dataset atual e maior dataset kdd99

4.3 Experimentos e Resultados

- gráficos, tempos, tabelas... - análises e comentários

Mostrar alguma implementação já feita e que esteja funcionando minimamente

Mostrar resultados mesmo que sejam bem simples e básicos, apenas para demonstrar que vc domina o ambiente e as ferramentas e que está apto a avançar no trabalho

passos feitos/a fazer 1. Entender Minas 2. Analisar/descrever dataset KDD 3. Notas sobre implementação Python/Kafka/Minas (não escala como esperado) 4. BigFlow (dataset mais novo, usa flink) 5. Plataforma Flink (processamento distribuído) Proposta 6. Implementar minas em Scala/Flink 7. Testar com datasets KDD e BigFlow 8. Validar/Comparar métricas com seus trabalhos correspondentes

- Descrição do hardware utilizado pode conter: - Arch, OS, Kernel, - CPU (core, thread, freq), - RAM (total/free size, freq), - Disk (total/free size, seq RW, rand RW), - Net IO between nodes (direct crossover, switched, wireless, to cloud) (bandwidth, latency). essas métricas permitem relacionar trade-offs para as questões de fog: Processar em node, edge ou cloud?

Provavelmente vou retirar o kafka da jogada em node/edge, deixando apenas em cloud.

Capítulo 5

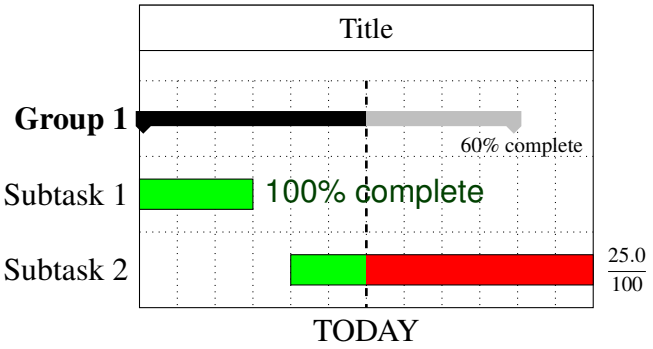
CRONOGRAMA

cap 5: Cronograma até a defesa

5.1 Cronograma

Lista de etapas

Quadro (linhas: etapas x colunas: meses)



Etapas previstas:

- E1 - Implementação distribuída do MINAS em uma rede de SBC com Rapberry: meses 01 a 03
- E2 - Preparação da base de dados para treinamento e teste: Mês 02
- E3 - Teste ... : meses 04 a 07
-

REFERÊNCIAS

BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. *Fog Computing and Its Role in the Internet of Things*. [s.n.], 2012. ISBN 9781450315197. Disponível em: ⟨http://www.lispmob.org⟩.

CASSALES, G. W.; SENGER, H.; DE FARIA, E. R.; BIFET, A. IDSA-IoT: An Intrusion Detection System Architecture for IoT Networks. In: *2019 IEEE Symposium on Computers and Communications (ISCC)*. [s.n.], 2019. p. 1–7. ISBN 978-1-7281-2999-0. ISSN 1530-1346. Disponível em: ⟨https://ieeexplore.ieee.org/document/8969609/⟩.

Dastjerdi, A. V.; Buyya, R. Fog computing: Helping the internet of things realize its potential. *Computer*, v. 49, n. 8, p. 112–116, Aug 2016. ISSN 1558-0814.

FARIA, E. R. de; CARVALHO, A. C. Ponce de L. F.; GAMA, J. Minas: multiclass learning algorithm for novelty detection in data streams. *Data Mining and Knowledge Discovery*, v. 30, n. 3, p. 640–680, May 2016. ISSN 1573-756X. Disponível em: ⟨https://doi.org/10.1007/s10618-015-0433-y⟩.

IEEE Communications Society. *IEEE Std 1934-2018 : IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing*. IEEE, 2018. 176 p. ISBN 9781504450171. Disponível em: ⟨https://ieeexplore.ieee.org/document/8423800⟩.

MELL, P.; GRANCE, T. The NIST definition of cloud computing: Recommendations of the National Institute of Standards and Technology. In: NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Public Cloud Computing: Security and Privacy Guidelines*. 2012. p. 97–101. ISBN 9781620819821. Disponível em: ⟨http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf⟩.

SHI, W.; CAO, J.; ZHANG, Q.; LI, Y.; XU, L. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, Institute of Electrical and Electronics Engineers Inc., v. 3, n. 5, p. 637–646, oct 2016. ISSN 23274662. Disponível em: ⟨https://ieeexplore.ieee.org/abstract/document/7488250⟩.