Construção de Modelos Baseados em *n*-gramas para Detecção de Anomalias em Aplicações Distribuídas

Amanda Viescinski, Tiago Heinrich, Newton C. Will, Carlos Maziero

Departamento de Informática
Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

{abviescinski,theinrich,ncwill,maziero}@inf.ufpr.br

Abstract. *Security is critical in distributed systems and applications. A common approach for security is intrusion detection, which can be performed by attack signatures or by anomaly detection. In the anomaly detection approach, a model of normal behavior of the system is built and then used to detect deviations in its behavior. This paper proposes a technique for building behavioral models of distributed applications using system logs from their nodes. Partial models are built based on sets of event n-grams, which are then combined to obtain more general models. The proposed technique was evaluated using logs obtained from a distributed file system, with promising results.*

Resumo. *A segurança é fundamental em sistemas distribuídos. Uma abordagem usual em segurança é a detecção de intrusão, que pode ser efetuada através da detecção de anomalias. Neste caso, um modelo de comportamento normal do sistema é construído e utilizado pelo sistema de detecção para checar desvios no comportamento do ambiente monitorado. Este artigo propõe uma técnica para a construção de modelos comportamentais de aplicações distribuídas através de traços de operação dos seus nós. São demonstrados os procedimentos realizados para a construção de modelos parciais, que são dispostos em conjuntos de n-gramas de eventos e combinados para obter modelos mais genéricos. Os resultados destacam a aplicação de um conjunto de dados real para a avaliação dos modelos, com resultados propícios na taxa de falso-positivo.*

1. Introdução

Nuvens computacionais são responsáveis por oferecer serviços através de diferentes formatos, que visam se adequar ao tipo de necessidade do seu cliente. Este cenário de sistemas distribuídos é responsável pela comunicação, gerenciamento de serviços e ferramentas através da troca de mensagens [Coulouris et al. 2013, Hauser et al. 2013]. Por consequência, o enfoque em garantir a segurança nesses ambientes é um fator de importância, com organizações utilizando *firewalls*, *anti-vírus* e sistemas de detecção de intrusão (*Intrusion Detection System* - IDS).

A detecção de intrusão é o mecanismo responsável pelo monitoramento de um conjunto de componentes, como aplicações em um *host* ou operações realizadas no ambiente de rede, sendo um enfoque recorrente na literatura [Mishra et al. 2017, Borkar et al. 2017, Jose et al. 2018, Khraisat et al. 2019]. Tal tarefa tem o intuito de reconhecer operações de natureza duvidosa ou indícios de possíveis incidentes

[Scarfone and Mell 2012]. Voltadas para um ambiente distribuído, abordagens padrões exploram um monitoramento local para os processos, armazenando eventos locais com um horário global para posterior análise em um sistema central [Lanoë et al. 2019].

A ordenação de eventos com base em um relógio global permite a análise de sequência de eventos para detectar comportamentos incorretos, podendo ser realizada com "assinaturas" de ataques. Para isso, é necessário uma base com os traços de execuções desses ataques, dando origem a uma assinatura de eventos que serão confrontadas com a sequência de eventos analisada [Totel et al. 2016]. Desta maneira, apenas ataques previamente conhecidos são detectados.

Ao explorar métodos de detecção por anomalia, a limitação para identificar ataques desconhecidos não está presente, já que a identificação está diretamente relacionada com a definição de um modelo de comportamento normal para o ambiente [Garcia-Teodoro et al. 2009]. Outro ponto relevante é a existência de um relógio global para a ordenação de todos os eventos que ocorreram no sistema, já que alcançar um alto nível de precisão entre os relógios locais vem a ser muito difícil, podendo ocasionar uma ordenação incorreta dos eventos.

A abordagem descrita neste trabalho permite a elaboração de uma nova estratégia para a identificação de anomalias em um sistema distribuído. A estratégia escolhida consiste da construção de modelos de comportamento normal para sistemas distribuídos, sem a necessidade de um relógio físico global. Isto é possível com a relação *happened-before* e atribuição de *timestamps* lógicos [Lamport 1978] que respeitam a ordem causal de cada evento ocorrido. Esta lista parcialmente ordenada de eventos globais permite a construção de conjuntos de n-gramas que irão compor os modelos de normalidade.

Este trabalho está estruturado em cinco seções. A Seção 2 descreve os principais conceitos para o estudo. Na Seção 3 são apresentados os trabalhos que têm sido realizados neste contexto. A técnica proposta para construção dos modelos é descrita na Seção 4. A Seção 5 avalia a técnica proposta e os resultados obtidos. Por fim, a Seção 6 traz as conclusões do trabalho.

2. Fundamentação Teórica

Esta seção apresenta os principais conceitos para o entendimento deste trabalho, como sistemas distribuídos e detecção de intrusão.

2.1. Sistemas Distribuídos

Um sistema distribuído se caracteriza pela utilização de diversos componentes, dispostos em diferentes computadores e interconectados através de redes de comunicação, que se comunicam e coordenam suas tarefas por meio da trocas de mensagens [Coulouris et al. 2013]. Consequentemente, tais sistemas oferecem amplo suporte à escalabilidade, já que componentes podem ser adicionados ou substituídos conforme a demanda. Neste contexto componentes podem ser equipamentos como computadores ou processos. Além disso, um sistema distribuído é capaz de suportar falhas independentes, utilizando mecanismos capazes de detectar, mascarar e contornar tais falhas. Outra característica é a transparência, através da ocultação da separação de seus componentes, sendo apresentado ao usuário como um sistema único, não como um conjunto de partes.

Um ponto central dos sistemas distribuídos é o compartilhamento de recursos, físicos e lógicos, que são disponibilizados a todos os processos que compõem o sistema. Tais processos executam um conjunto de ações para a realização de tarefas, e cada ação individual é denominada como *evento*, com vários eventos acontecendo em cada processo. Por fim, a coordenação de tarefas em sistemas distribuídos geralmente depende de uma noção compartilhada de tempo. Devido às características de arquitetura e comunicação, em sistemas distribuídos não há uma noção única e global de tempo, sendo necessária a utilização de algoritmos específicos para determinar a ordenação de eventos.

2.2. Ordenação de Eventos

A ordenação de eventos ocorridos em um único processo é facilmente obtida através da utilização do relógio local, mas, quando se trata de sistemas distribuídos, essa ordenação não é trivial. Como não é possível alcançar uma sincronia entre todos os relógios de um sistema distribuído, se torna inviável utilizar o tempo físico para obter a ordenação dos eventos globais. Desta forma, técnicas que não tem a necessidade do uso do tempo físico destacam vantagens, como "aconteceu antes de" (ou "*happened before*"), denotada como $\rightarrow$, que foi definida por [Lamport 1978] para obter uma ordenação parcial dos eventos.

A relação "*happened before*", no conjunto de eventos de um sistema, é a menor relação que satisfaz as seguintes condições:

(I) Se a e b são eventos que ocorrem em um mesmo processo, usando o relógio local é possível inferir que a aconteceu antes de b, logo $a \rightarrow b$;
(II) Se $send(m)$ corresponde ao envio de uma mensagem m, e $receive(m)$ é a recepção da mensagem em outro processo, então $send(m) \rightarrow receive(m)$;
(III) Se $a \rightarrow b$ e $b \rightarrow c$, então $a \rightarrow c$.

Através da relação $\rightarrow$ é possível definir uma ordem parcial do sistema, porém ela não é suficiente para encontrar a ordem total (global), pois existem eventos concorrentes em que não se pode afirmar que $a \rightarrow b$ ou $b \rightarrow a$. Tal condição ocorre na presença de eventos em diferentes processos em que não há encadeamento de mensagens entre eles, sendo tais eventos denotados como $a||b$. Entretanto, há uma extensão desta abordagem, chamada *relógios lógicos*, que permite alcançar uma ordenação total válida dos eventos [Lamport 1978].

O conceito de relógio lógico é baseado na existência de contadores locais (internamente a cada processo) que permitem atribuir *timestamps* (ou carimbos de tempo) aos eventos que ocorreram em diferentes processos do sistema distribuído. Este contador local é incrementado monotonicamente e não possui nenhuma ligação em particular com qualquer relógio físico. Este incremento é feito no contador de cada processo p_i através de uma função C_i, presente em todos os processos, que atribui um valor inteiro $C_i(a)$ para todo evento a que ocorreu no processo p_i. Para capturar as relações entre os eventos, os processos atualizam seus respectivos relógios lógicos e transmitem seus valores nas mensagens trocadas entre si, de acordo com as seguintes regras:

(I) O processo p_i incrementa seu contador local ($C_i = C_i + 1$);
(II) O processo p_i envia uma mensagem m acompanhada de um *timestamp* t, indicando o valor de seu contador local C_i ($send(m, t)$);
(III) O processo p_j recebe a mensagem ($receive(m, t)$), atualiza o valor de seu contador local aplicando a regra $C_j = max(C_j, t)$, aplica a regra (I) e assinala o resultado como indicativo de tempo para o recebimento da mensagem.

2.3. Detecção de Intrusão

Detecção de intrusão é o processo de monitoramento das redes e de seus componentes, como computadores ou aplicações, com o propósito de reconhecer atividades de natureza duvidosa ou indícios de possíveis incidentes [Scarfone and Mell 2012, Angiulli et al. 2015].

Sistemas de detecção de intrusão, ou *Intrusion Detection System (IDS)*, são responsáveis por efetuar o processo de detecção. Estes podem ser baseados em assinaturas, que consistem em bases de dados de comportamentos reconhecidos como ataques [Debar et al. 1999, Yassin et al. 2013]. Embora seja eficiente para detectar ataques conhecidos, esse tipo de detecção acaba sendo ineficiente para ataques ainda não conhecidos.

Outra possível abordagem é a detecção por anomalia (ou Anomaly-based Detection - AD), que utiliza perfis de comportamentos normais do sistema. Uma base de traços do sistema é responsável pela representação deste comportamento normal. Portanto, um IDS realiza a análise dos dados e construção de um modelo de comportamento normal, após o período de aprendizado, o comportamento do sistema é confrontado a esse modelo, em busca de comportamentos anormais [Zolotukhin and Hämäläinen 2013]. Assim, qualquer desvio desses modelos é considerado um ataque.

2.4. n-gramas

A detecção por anomalia aplica métodos de aprendizagem que dependem de dados numéricos como vetores de características para treinamento dos seus classificadores [Wressnegger et al. 2013]. Uma opção utilizada para a obtenção desses dados explora métodos de n-gramas, onde um n-grama consiste em um fragmento de n itens consecutivos (letras, palavras, mensagens, eventos, etc.) [Zolotukhin and Hämäläinen 2013].

Formalmente, dada uma *string* X que contém uma sequência de símbolos pertencente a um alfabeto A, obtém-se o conjunto de n-gramas de X lendo-se X sequencialmente com uma janela de comprimento n. O tamanho da janela pode variar de acordo com a estratégia utilizada ou tipo de dado. Para o respectivo trabalho considera-se como n-grama uma sequência de n eventos consecutivos no sistema, que podem ou não ocorrer no mesmo nó.

A sequência de eventos usada para a extração dos n-gramas é denominada um *traço* do sistema e representa uma execução de uma aplicação distribuída. Assim, quando a aplicação é executada um conjunto de vezes (1, 2, .., i) para cada execução um novo traço é gerado (t_1, t_2, .., t_i). Esta representação de traços t_i é similar a um conjunto de *logs*, aplicados para definir um ou mais modelos. É importante destacar que durante a fase de coleta de traços de comportamento normal é assumido que nenhum comportamento anormal ou ataque tenha ocorrido [Lanoë et al. 2019].

3. Trabalhos Relacionados

Visando a garantia de qualidade de serviço através da detecção de anomalias em um sistema distribuído, [Fu et al. 2009] propõem uma ferramenta para realizar o monitoramento dos *logs* e efetuar a identificação de anomalias. Uma máquina de estados finita[1] (Finite-

[1]Estruturas que contêm um número finito de estados e produzem saídas nas transições de estados após o recebimento de entradas [Lee and Yannakakis 1996].

State Automaton - FSA), é utilizada para determinar um modelo de comportamento de execução para cada sistema. Este conjunto finito de estados representa o comportamento normal do sistema através da extração e filtragem dos *logs*. A solução explora a utilização de *logs* oferecendo uma ferramenta que realiza a leitura de informações não estruturadas encontradas com frequência em um ambiente distribuído e a extração dos dados relevantes.

A metodologia publicada em [Totel et al. 2016] resulta em um modelo de comportamentos de aplicações distribuídas, tendo como base um autômato formando um cálculo distribuído e uma lista de propriedades temporais que este cálculo deve desempenhar, utilizando apenas a ordenação parcial dos eventos. Para que o modelo fosse criado, uma aplicação distribuída foi executada, gerando traços de eventos durante várias execuções. Relógios vetoriais foram utilizados para realizar a ordenação dos eventos. Na fase inicial, *logs* resultantes dos rastreamentos coletados são analisados para se obter uma visão global do comportamento. Em seguida, todos os comportamentos são combinados transformando-se em um único autômato global. Para que novos comportamentos fossem inferidos no modelo e retiradas eventuais atividades incorretas, foram implementadas propriedades temporais que devem ser processadas pelo sistema durante sua execução. Essas propriedades são derivadas dos rastreamentos e devem ser preservadas somente as que são válidas para todos esses traços. Para efetivar a validação do modelo um verificador foi utilizado.

O modelo presente em [Lanoë et al. 2019] é um aprimoramento da abordagem utilizada por [Totel et al. 2016], tendo como diferencial o tempo de processamento para produzir o modelo inicial e a etapa de generalização. Para reduzir o tempo gasto para a preparação do primeiro modelo, foi aplicado o conceito de corte consistente, que resulta em um reticulado (*Lattice*), com base em uma seleção rigorosa de determinadas sequências que possibilitam alcançar um modelo mais compacto, mantendo sua eficiência. Com isso, permitiu-se que um modelo iterativo fosse apresentado, onde o protótipo em construção é continuamente atualizado e generalizado.

Buscando realizar a identificação de falhas ou *bugs* através do monitoramento de traços, [Jiang et al. 2006] combina a variação no comprimento de n-gramas com a frequência de ocorrência dos mesmos. Após o agrupamento dos traços com seus respectivos grupos de n-gramas do mesmo tamanho, autômatos são construídos e utilizados para realizar a identificação de traços anormais.

Um olhar detalhado da detecção de intrusão utilizando n-gramas é apresentado em [Wressnegger et al. 2013], discutindo o uso de técnicas de detecção, classificação de anomalias e critérios para a seleção dos dados a serem utilizados na fase de aprendizado. O trabalho foca em oferecer informações que viabilizam o entendimento sobre a criação de modelos com n-gramas, apresentando critérios que permitem a avaliação da complexidade da detecção.

4. Proposta

Os IDSs têm sido amplamente empregados para automatizar o processo de detecção de intrusão. Dentre as técnicas de detecção utilizadas, a abordagem por anomalia apresenta maior eficácia para identificar novos tipos de ataques, quando comparada com as baseadas em assinatura e em análise do estado do protocolo [Liao et al. 2013]. Frequentemente,

o conceito de n-gramas é aplicado na construção de modelos de normalidade comportamental do sistema, que possibilitam a detecção de intrusão com base em anomalias [Wressnegger et al. 2013]. Entretanto, apesar de atualmente existirem diversas propostas de sistemas de detecção de intrusão em aplicações distribuídas baseados em anomalias [Stillerman et al. 1999, Totel et al. 2016, Lanoë et al. 2019], não identificamos nenhum trabalho publicado aplicando técnicas baseadas em n-gramas.

No contexto da detecção de intrusão baseada em anomalias, o uso de n-gramas para modelar o comportamento normal de aplicações distribuídas pode aumentar a agilidade no processo de detecção, além de proporcionar uma redução da quantidade de alertas gerados, aumentando precisão de IDSs para esse tipo de sistema. Assim, o objetivo do presente trabalho é a construção de modelos de comportamentos normais de aplicações distribuídas através do uso de n-gramas, para sistemas de detecção de intrusão baseados em anomalias.

As notações necessárias para o entendimento dos modelos descritos são apresentadas na Tabela 1. Este conjunto de representações é aplicado no decorrer de todo o trabalho.

Tabela 1. Notações utilizadas neste trabalho.

Notação	Descrição
n	*tamanho de uma grama*
t	*um traço*
M	*um modelo*
$M(n)$	*modelo com gramas de tamanho* n
$G(t,n)$	*conjunto de* n*-gramas de tamanho* n *presentes em* t
TC	*um conjunto de traços corretos*
SC	*subconjuntos de* TC
SC_i	*um subconjunto de* TC
v	*quantidade de subconjuntos/modelos*
MV	*modelos de validação diversificado de* SC
MV_i	*um modelo de validação*
$MV\cup$	*todos os modelos de união pertencentes a* MV
$MV\cap$	*todos os modelos de interseção pertencentes a* MV
$MV\cup_i$	*um modelo de união*
$MV\cap_i$	*um modelo de interseção*

4.1. Obtenção dos n-gramas

Para construir um modelo M que represente o comportamento normal do sistema, é realizado o particionamento dos eventos ordenados de cada execução em conjuntos de n-gramas únicos. Um modelo $M(n)$ é um conjunto de n-gramas de tamanho n que representa o comportamento normal do sistema. Este é gerado a partir de um conjunto de traços corretos (sem ataque ou quaisquer anomalias) obtidos do sistema. Intuitivamente, quanto mais traços corretos forem considerados, mais comportamentos corretos serão representados pelo modelo M, levando a menos falso-positivos (eventos normais erroneamente vistos como ataques).

A Figura 1 apresenta um exemplo de execução entre dois clientes e um servidor, contendo todos os eventos (identificados pelas letras) e duas possíveis sequências com

4-gramas que podem ser capturadas com essa execução (*abcd* e *mghi*). Diversas outras sequências de eventos podem ser formadas com 4-gramas, ou usando outros valores para n.

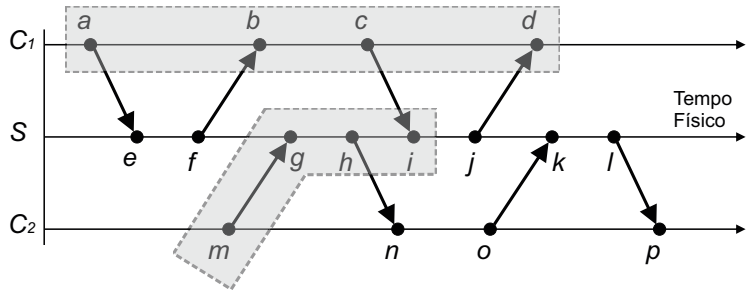

Figura 1. Exemplo de n-gramas de uma execução.

A representação do conjunto com todos os n-gramas de tamanho n presentes em um traço t é dada por $G(t, n)$. Esse conjunto é obtido através de um algoritmo de busca em profundidade no traço t. Tomando como exemplo a Figura 1, a partir do evento *a* é possível construir quatro conjuntos com 4-gramas: *abcd*, *aefb*, *abci* e *aefg*. Da mesma forma, tomando o evento *m* como início, é possível construir os conjuntos 4-gramas *mghn*, *mnok* e *mnop*, além do conjunto *mghi* já destacado na figura. Na execução da Figura 1 existem ao todo 27 n-gramas distintos de tamanho 4.

A determinação da ordem dos eventos é realizada utilizando o algoritmo proposto em [Lamport 1978], onde a ordem causal é reconstruída através da atribuição dos eventos utilizando um contador. Ao implementar este algoritmo, os eventos que ocorreram no sistema (locais e globais) são ordenados seguindo a causalidade entre eles através do uso de *timestamps*, que permite gerar uma ordenação completa mais tênue e fácil de ser observada para cada execução da aplicação.

4.2. Modelos de Comportamento Normal

A construção apropriada de um modelo depende da garantia de que os traços utilizados estejam livres de ataques ou de qualquer tipo de anomalia. Sendo TC um conjunto de traços corretos obtidos do sistema, com cada traço $t \in TC$ é possível obter um conjunto de n-gramas $G(t, n)$, que é uma representação parcial do comportamento correto.

Inicialmente, TC é dividido em v subconjuntos $SC_i, i = 1 \ldots v$ independentes, que serão utilizados tanto na construção dos modelos quanto para avaliar o processo de detecção de anomalias. Esses subconjuntos devem ser proporcionais em termos de quantidade e tamanho de traços. Para nossas avaliações consideramos $v = 5$. Buscando aumentar a eficiência do algoritmo de criação dos modelos, foi criado um arquivo único contendo todas as gramas encontradas nos traços que compõem cada subconjunto, ou seja, implicitamente está sendo usada a união para combinar esses resultados.

A seguir, são gerados v modelos de validação MV_i, cada um usando $v - 1$ subconjuntos, em um esquema de validação cruzada *K-fold*. Para isso, os subconjuntos são combinados de duas formas, por união: $MV_i(SC, n) = \cup_i SC_i(t, n)\ \forall t \in TC$ e por interseção: $MV_i(SC, n) = \cap_i SC_i(t, n)\ \forall t \in TC$.

O uso do modelo $M(n)$ na detecção de anomalias é simples: se um traço t somente contiver n-gramas presentes em $M(n)$, t é considerado normal perante $M(n)$; se o traço

t contiver n-gramas não-presentes em $M(n)$, t é considerado suspeito perante $M(n)$. Matematicamente, se $G(t,n) \subseteq M(n)$ então t é normal perante $M(n)$; senão, t é suspeito perante $M(n)$. Para a detecção de anomalias *online*, ou seja, durante a execução do sistema, n-gramas individuais podem ser construídos e testados dinamicamente: um n-grama g é considerado normal se $g \in M(n)$, e suspeito caso contrário.

Outra forma de realizar a detecção é utilizar os subconjuntos em uma estratégia de votação [Raguenet and Maziero 2008]: um traço t é confrontado aos $SC(n)$; se for aceito pela maior parte dos subconjuntos (ou por um número mínimo deles), t é considerado normal (não-suspeito).

5. Avaliação Experimental

A avaliação da proposta consiste na construção de um conjunto de modelos a partir de dados de uma aplicação distribuída real. A base de dados usada no estudo é constituída de *logs* de uma aplicação distribuída e foi apresentada em [Lanoë et al. 2019]. Essa base ainda não é pública, mas foi escolhida por apresentar múltiplos conjuntos de execuções de uma aplicação real, que são representados por traços dos clientes.

5.1. Cenário e Método de Coleta dos Traços

O cenário de avaliação usa como aplicação distribuída o *XtreemFS*, que provê um sistema de arquivos distribuído, replicado e tolerante a falhas [Quobyte Inc 2020]. Para fornecer seus serviços são necessários quatro componentes: OSD (*Object Storage Device*), entidade encarregada de armazenar o conteúdo real dos arquivos; MRC (*Metadata and Replica Catalog*), que armazena os metadados dos arquivos, gerencia a árvore de diretórios e realiza o controle de acesso; DIR (*Directory Service*), responsável por conectar todos os demais componentes e registrar os serviços do sistema, este componente inclui um banco de dados e volumes de armazenamento; por fim, os clientes que realizam requisições ao sistema e gerenciam os estados dos arquivos, executando operações como criação/montagem de volumes, criação de arquivos, etc.

A base usada na avaliação contém dados obtidos através de 126 execuções, cada uma contendo os traços de todos os nós do sistema: OSD, MRC, DIR e dois clientes. Essas execuções representam situações de comportamento normal do sistema e também do sistema sob ataque, com durações que vão de 1 minuto a 5 horas. Os traços tem um tamanho médio de 39 KB, e cerca de 16,3% dos traços da base representam comportamento anormal.

Para a obtenção da base, todos os nodos do sistema foram configurados para registrar seus traços individuais, contendo informações sobre eventos de envio ou recepção de mensagens. Cada evento registrado contém os seguintes campos: *Source IP* armazena o endereço IP do nodo que deu origem ao evento; *Destination IP* indica o endereço do nodo de destino; *Type* informa se a mensagem corresponde a um envio (!) ou recepção (?); o último campo, *Data*, sinaliza a natureza do evento, podendo ser usado para identificar se determinado evento é uma requisição ou uma resposta. A Figura 2 apresenta a estrutura desses registros para uma execução hipotética entre dois clientes e um servidor, os eventos são os mesmos demonstrados na Figura 1.

Objetivando coletar traços de execuções com comportamentos anômalos do sistema, uma versão não segura do XtreemFS foi utilizada para que quatro ataques conhe-

Nó "servidor" (IP1)

```
IP2 - IP1 ? consulta
IP3 - IP1 ? consulta
IP1 - IP2 ! disponível
IP1 - IP3 ! disponível
IP2 - IP1 ? adquirir
IP3 - IP1 ? adquirir
IP1 - IP2 ! venda
IP1 - IP3 ! venda
```

Nó "cliente c_1" (IP2)

```
IP2 - IP1 ! consulta
IP1 - IP2 ? disponível
IP2 - IP1 ! adquirir
IP1 - IP2 ? venda
```

Nó "cliente c_2" (IP3)

```
IP3 - IP1 ! consulta
IP1 - IP3 ? disponível
IP3 - IP1 ! adquirir
IP1 - IP3 ? venda
```

Figura 2. Exemplo de traços de uma aplicação distribuída.

cidos contra a sua integridade pudessem ser realizados: o ataque *NewFile*, que adiciona metadados de um arquivo no servidor MRC e não insere seu conteúdo no servidor OSD; *DeleteFile*, que exclui os metadados do arquivo no servidor MRC e mantém seu conteúdo no servidor OSD; *Chmod*, que altera a política de acesso a arquivos no servidor MRC, mesmo sem a autorização; e, por último, o ataque *Chown*, que modifica o proprietário do arquivo nos metadados MRC.

Cada um dos ataques foi executado em quatro contextos diferentes: (c1) nenhum cliente ativo; (c2) com clientes ativos no ambiente, mas antes de realizarem operações; (c3) após as operações serem realizadas e; (c4) originado de um endereço que não pertence ao cliente. Os traços dessas execuções permitem a avaliação da eficácia de detecção de cada modelo previamente construído a partir do comportamento normal do sistema.

5.2. Modelos de Normalidade

Modelos de normalidade são utilizados por sistemas de detecção baseados em anomalias para identificar se uma execução é normal ou anômala. Assim, é produzido um alerta caso um comportamento que não esteja contido nestes modelos seja detectado. Quando um alerta gerado sinaliza erroneamente um comportamento normal como uma anomalia, este alerta é considerado um falso-positivo (FP). Neste sentido, o primeiro experimento visa avaliar a representatividade dos modelos propostos na Seção 4.2, através da quantificação de FPs.

Para avaliar os modelos de união $MV\cup$ e interseção $MV\cap$, cada MV_i é testado usando todos os traços corretos $TC_i \in SC_i$ que não foram adotados para compor MV_i. Em uma primeira abordagem, se $G(t,n) \subseteq MV_i(n), \forall t \in TC_i$ então MV_i está correto e TC_i não representa um falso-positivo. Entretanto, há uma probabilidade significativa de que os modelos criados não sejam totalmente representativos, ou seja, não possuam todos os comportamentos normais da aplicação.

Como a relação $\subseteq$ define de forma binária o resultado (normal ou anormal), se torna necessário adotar um critério menos rígido. Por isso, foi adotada uma *taxa de aceitação* para flexibilizar a indicação de comportamentos normais e anômalos. Assim, nesta avaliação, foi considerado como resultado retornado pelo teste de cada MV_i o percentual de n-gramas encontrados nos modelos em relação ao total de n-gramas presentes no traço analisado. Como critério de normalidade, foi estipulado que um traço é considerado correto pelo $MV\cup$ se contiver no mínimo 95% de gramas conhecidas, enquanto para o $MV\cap$ o traço deve conter pelo menos 80% de gramas conhecidas.

A Tabela 2 apresenta a saída obtida nos testes de cada $TC_i \in SC_i$ para $MV\cup$ e $MV\cap$. Cada resultado representa a média do percentual de n-gramas obtidos dos traços pertencentes a SC_i presentes no respectivo modelo. Os modelos foram avaliados com diversos valores de n, para estudar a influência do tamanho dos n-gramas nos resultados. Contudo, a diferença dos resultados não foi significativa para todos os valores de n testados; desta forma, somente os casos relevantes (3-gramas e 12-gramas) são apresentados.

Tabela 2. Média dos percentuais de n-gramas de cada $TC_i \in SC_i$ encontradas nos respectivos MV_i.

		Subconjunto				
N	**Modelo**	SC_1	SC_2	SC_3	SC_4	SC_5
3	$MV\cup$	100%	99,9%	99,9%	99,9%	99,7%
	$MV\cap$	97,1%	96,6%	97,2%	96,8%	96,7%
12	$MV\cup$	99,8%	98,9%	99,4%	99,6%	97,3%
	$MV\cap$	74,2%	73,2%	71,6%	92,3%	72,6%

Ao analisar a variação dos resultados em relação ao tamanho n das gramas, observa-se que o modelo de união $MV\cup$ é pouco sensível à variação de n, levando a variações na taxa de acerto inferiores a 3%. Por outro lado, o modelo de interseção $MV\cap$ é bastante sensível ao valor n, apresentado uma variação de até 25% nas taxas de acerto entre 3-gramas e 12-gramas. Esse impacto de n no método de avaliação ocorre devido ao crescimento no tamanho dos gramas; basicamente, quanto maior o n-grama, menos frequente ele se torna e menor a sua probabilidade de aparecer na interseção dos subconjuntos.

Desta forma, o tamanho dos n-gramas tem menos impacto no modelo de união. Apesar do nível de detecção dos modelos estar com uma média de acertos acima de 70%, o modelo de união está acima do esperado em relação aos 95% de taxa de aceitação e o modelo de interseção esta 9 pontos percentuais abaixo dos 80% de taxa de aceitação definidos, considerando o pior caso.

Apesar do crescimento no tamanho dos n-gramas colaborar para representação de eventos em um ambiente, como relatado nos modelos da Tabela 2, é possível identificar a dificuldade que estas representações mais longas apresentam para um modelo. Isso se deve ao fato de que modelos como o de interseção seriam prejudicados caso estas ocorrências não existissem no conjunto de dados avaliados.

Tabela 3. Número médio de n-gramas obtidos dos traços, em função de n.

Valor de n	3	6	9	12
Número de n-gramas	970	2.790	7.369	20.980

A Tabela 3 informa o número médio de n-gramas obtidos dos traços de execuções. Observa-se claramente que o número de n-gramas cresce com o valor de n, em uma razão sub-exponencial. Portanto, modelos gerados com n-gramas mais longos são maiores (têm mais n-gramas), contudo representam comportamentos mais específicos, conforme apontado pelos resultados da Tabela 2.

Finalmente, a Tabela 4 traz uma avaliação individual dos modelos produzidos por cada subconjunto SC_i. Cada entrada m_{ij} da matriz indica o percentual de traços de SC_j aceitos pelo modelo produzido a partir dos traços de SC_i para $n = 12$. Essa tabela permite então estimar o quão representativa é a contribuição de cada subconjunto para a construção dos modelos MV. Na tabela equivalente para $n = 3$ todos os valores são 100%.

Tabela 4. Percentual de traços de um subconjunto aceitos pelos outros subconjuntos, para $n = 12$.

	Subconjunto				
Modelo	SC_1	SC_2	SC_3	SC_4	SC_5
SC_1	100%	72,2%	84,2%	68,4%	63,2%
SC_2	52,6%	100%	57,9%	84,2%	52,6%
SC_3	84,2%	72,2%	100%	73,7%	89,5%
SC_4	63,2%	83,3%	63,2%	100%	78,9%
SC_5	68,4%	72,2%	78,9%	73,7%	100%

Os subconjuntos com melhor representação são SC_2 e SC_4, o subconjunto SC_2 é o que melhor consegue representar outros modelos, mas é o modelo menos representado pelos demais subconjuntos. Portanto, a Tabela 4 demonstra a distribuição que existe entre os traços e os subconjuntos, deixando evidente o impacto que um subconjunto possui nos modelos de validação MV e destacando a dificuldade de representação que a base possui.

5.3. Detecção de Ataques

A avaliação de detecção de ataques depende da estimativa de verdadeiro-positivo pelo modelo. Esta técnica consiste em testar se cada modelo MV_i é capaz de detectar cada ataque a_i no cenário c_i considerando o mesmo conjunto de avaliação apresentado na Seção 5.2.

A Tabela 5 apresenta a relação de verdadeiro-positivo para os modelos baseados em união e em interseção, para n-gramas de tamanho 3 e 12. Como apresentado na Seção 5.1, os quatro tipos de ataques representados na base de dados são considerados. Na tabela, uma indicação $x/5$ indica que x dos 5 modelos detectaram o ataque; por clareza, $0/5$ é indicado como "-".

Tabela 5. Detecção de verdadeiro-positivo por cada modelo.

	Ataque	*NewFile*				*DeleteFile*				*Chmod*				*Chown*			
N	**Contexto**	c1	c2	c3	c4	c1	c2	c3	c4	c1	c2	c3	c4	c1	c2	c3	c4
3	$MV\cup$	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	$MV\cap$	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
12	$MV\cup$	-	2/5	5/5	-	-	-	1/5	-	-	5/5	5/5	-	-	-	-	1/5
	$MV\cap$	-	5/5	-	5/5	5/5	5/5	2/5	5/5	5/5	5/5	-	4/5	-	4/5	-	4/5

A Tabela 5 mostra que não há uma diferença clara de eficiência entre os modelos de união e de interseção na detecção de ataques, sobretudo para valores de n pequenos. Em particular, observa-se que com $n = 3$ nenhum ataque foi identificado.

Já para $n = 12$, o nível de identificação ainda é baixo, mas melhor que na situação anterior. Relacionando os dois conjuntos de dados do modelo de normalidade com os modelos de ataques, é possível concluir que valores maiores de n permitem a identificação de ataques com mais facilidade mas, em contrapartida, aumentam a dificuldade de identificar corretamente os traços normais.

A dificuldade de representar corretamente o comportamento normal do sistema com n-gramas mais longos, possivelmente está relacionada com o tamanho da base de traços utilizada. É provável que os traços usados para a construção dos modelos não contenham um número suficiente de comportamentos normais para poder representar adequadamente o sistema.

Soluções para esse problema implicariam em a) obter mais traços de comportamento normal e de ataques para enriquecer os modelos; e b) aumentar o poder de expressão dos modelos, através da generalização dos n-gramas, de forma similar à efetuada para os autômatos de estados finitos na abordagem descrita em [Lanoë et al. 2019].

6. Conclusão

Este trabalho apresentou uma técnica para a construção de modelos de normalidade de uma aplicação distribuída utilizando como base n-gramas de eventos. Foram descritos os processos para a obtenção dos n-gramas através dos traços de execução de uma aplicação distribuída e a construção de modelos de normalidade do sistema, através de operações sobre os conjuntos.

A avaliação experimental apresentou resultados distintos na identificação de traços normais e de ataques, sendo influenciados pelo tamanho dos n-gramas. Foram avaliadas duas estratégias de construção de modelos de normalidade (união e interseção) em quatro cenários de ataque distintos. Os resultados obtidos mostram que os modelos baseados em união oferecem uma boa capacidade de modelar o comportamento normal, mas não têm boa capacidade de detecção de ataques. No sentido contrário, os modelos baseados em interseção são mais sensíveis para detectar ataques, mas não modelam bem a normalidade. Além disso, observa-se que n-gramas mais longos detectam ataques mais facilmente, porém são mais propensos a falso-positivos.

Este projeto contempla diversos trabalhos futuros, alguns já em andamento. Estamos trabalhando para aumentar o número de execuções normais e de ataques na base de traços, para ter uma representação mais ampla e equilibrada dos comportamentos possíveis. Outra frente em andamento consiste em introduzir a noção de generalização na representação dos n-gramas, a fim de obter uma representação mais abrangente (e ao mesmo tempo mais compacta) do comportamento normal do sistema. Por fim, estratégias mais complexas de combinação dos resultados dos modelos parciais (além de união e interseção) também estão sob estudo, em particular estratégias baseadas em lógica difusa [Raguenet and Maziero 2008].

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001. Os autores também agradecem aos departamentos de Informática da UFPR e UTFPR.

Referências

Angiulli, F., Argento, L., and Furfaro, A. (2015). Exploiting n-gram location for intrusion detection. In *Proceedings of the 27th International Conference on Tools with Artificial Intelligence*, pages 1093–1098, Vietri sul Mare, Itália. IEEE.

Borkar, A., Donode, A., and Kumari, A. (2017). A survey on intrusion detection system (IDS) and internal intrusion detection and protection system (IIDPS). In *Proceedings of the International Conference on Inventive Computing and Informatics*, pages 949–953, Coimbatore, Índia. IEEE.

Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. (2013). *Sistemas Distribuídos-: Conceitos e Projeto*. Bookman Editora.

Debar, H., Dacier, M., and Wespi, A. (1999). Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822.

Fu, Q., Lou, J.-G., Wang, Y., and Li, J. (2009). Execution anomaly detection in distributed systems through unstructured log analysis. In *Proceedings of the 9th International Conference on Data Mining*, pages 149–158, Miami, FL, EUA. IEEE.

Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18–28.

Hauser, C., Tronel, F., Fidge, C., and Mé, L. (2013). Intrusion detection in distributed systems, an approach based on taint marking. In *Proceedings of the International Conference on Communications*, pages 1962–1967, Budapeste, Hungria. IEEE.

Jiang, G., Chen, H., Ungureanu, C., and Yoshihira, K. (2006). Multiresolution abnormal trace detection using varied-length n-grams and automata. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(1):86–97.

Jose, S., Malathi, D., Reddy, B., and Jayaseeli, D. (2018). A survey on anomaly based host intrusion detection system. *Journal of Physics: Conference Series*, 1000:012049.

Khraisat, A., Gondal, I., Vamplew, P., and Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1).

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565.

Lanoë, D., Hurfin, M., Totel, E., and Maziero, C. (2019). An efficient and scalable intrusion detection system on logs of distributed applications. In *Proceedings of the International Conference on ICT Systems Security and Privacy Protection*, pages 49–63, Lisboa, Portugal. Springer.

Lee, D. and Yannakakis, M. (1996). Principles and methods of testing finite state machines - a survey. *Proceedings of the IEEE*, 84(8):1090–1123.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., and Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24.

Mishra, P., Pilli, E. S., Varadharajan, V., and Tupakula, U. (2017). Intrusion detection techniques in cloud environment: A survey. *Journal of Network and Computer Applications*, 77:18 – 47.

Quobyte Inc (2020). XtreemFS - fault-tolerant distributed file system. `http://www.xtreemfs.org`.

Raguenet, I. and Maziero, C. (2008). A fuzzy model for the composition of intrusion detectors. In *Proceedings of the 23rd International Information Security Conference*, pages 237–251, Milão, Itália. Springer.

Scarfone, K. and Mell, P. (2012). Guide to intrusion detection and prevention systems (IDPS). Technical report, National Institute of Standards and Technology.

Stillerman, M., Marceau, C., and Stillman, M. (1999). Intrusion detection for distributed applications. *Communications of the ACM*, 42(7):62–69.

Totel, E., Hkimi, M., Hurfin, M., Leslous, M., and Labiche, Y. (2016). Inferring a distributed application behavior model for anomaly based intrusion detection. In *Proceedings of the 12th European Dependable Computing Conference*, pages 53–64, Gotemburgo, Suécia. IEEE.

Wressnegger, C., Schwenk, G., Arp, D., and Rieck, K. (2013). A close look on n-grams in intrusion detection: anomaly detection vs. classification. In *Proceedings of the Workshop on Artificial Intelligence and Security*, pages 67–76, Berlim, Alemanha. ACM.

Yassin, W., Udzir, N. I., Muda, Z., Sulaiman, M. N., et al. (2013). Anomaly-based intrusion detection through k-means clustering and naives bayes classification. In *Proceedings of the 4th International Conference on Computing and Informatics*, pages 298–303, Kuching, Malásia.

Zolotukhin, M. and Hämäläinen, T. (2013). Detection of anomalous HTTP requests based on advanced n-gram model and clustering techniques. In *Proceedings of the 13th International Conference on Internet of Things, Smart Spaces, and Next Generation Networking*, pages 371–382. Springer, São Petersburgo, Rússia.