Universidade Estadual de Londrina

SEAN CARLISTO DE ALVARENGA

ANÁLISE DE ALERTAS DE INTRUSÃO BASEADA EM MINERAÇÃO DE PROCESSOS

Londrina
2016

SEAN CARLISTO DE ALVARENGA

ANÁLISE DE ALERTAS DE INTRUSÃO BASEADA EM MINERAÇÃO DE PROCESSOS

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Bruno Bogaz Zarpelão.

Londrina
2016

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Alvarenga, Sean Carlisto de.
Análise de Alertas de Intrusão Baseada em Mineração de Processos / Sean Carlisto de Alvarenga. - Londrina, 2016.
99 f. : il.

Orientador: Bruno Bogaz Zarpelão.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2016.
Inclui bibliografia.

1. Detecção de intrusão - Teses. 2. Visualização de segurança - Teses. 3. Mineração de alertas - Teses. I. Zarpelão, Bruno Bogaz . II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. III. Título.

SEAN CARLISTO DE ALVARENGA

ANÁLISE DE ALERTAS DE INTRUSÃO BASEADA EM MINERAÇÃO DE PROCESSOS

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

_____

Orientador: Prof. Dr. Bruno Bogaz Zarpelão
Universidade Estadual de Londrina – UEL

_____

Prof. Dr. Sylvio Barbon Junior
Universidade Estadual de Londrina – UEL

_____

Prof. Dr. Mario Lemes Proença Jr.
Universidade Estadual de Londrina – UEL

_____

Prof. Dr. Alexandre de Aguiar Amaral
Instituto Federal de Santa Catarina - IFSC

Londrina, 06 de Abril de 2016.

AGRADECIMENTOS

A Deus.

Aos meus pais e ao meu irmão. Em especial a minha mãe Vera, que sempre me apoiou e incentivou e não mediu esforços para que os meus estudos fossem prioridade. Muito obrigado a todos por todo apoio.

Aos meus amigos, pelas conversas, pelos momentos de alegria e descontração. Em especial: Vinícius, Gustavo, Guilherme, Jonas, Letícia, Cachopo, Pedrão e Igawa.

Ao meu orientador Prof. Dr. Bruno Bogaz Zarpelão, que me acompanha desde meu TCC, agradeço por sua orientação, pela paciência, pelo apoio, por todas as oportunidades e pela confiança em meu trabalho. Sem dúvidas esse foi um período de muito aprendizado que vou carregar pro resto da vida.

Ao Prof. Dr. Sylvio Barbon Junior, pelas reuniões, por suas ótimas ideias, pelas discussões e sugestões que ajudaram a resolver os problemas que surgiram ao longo deste trabalho.

Ao Prof. Dr. Rodrigo Sanches Miani e a Universidade de Maryland, por disponibilizar e permitir o uso das informações utilizadas neste trabalho.

Aos membros da banca examinadora por todas as dicas e sugestões que ajudaram no desenvolvimento e aperfeiçoamento deste trabalho.

Ao Universidade Estadual de Londrina (UEL) e ao Departamento de Computação (DC) por prover um ótimo ambiente e infraestrutura para seus estudantes realizarem suas pesquisas.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte financeiro durante o programa que permitiu a realização deste trabalho.

ALVARENGA, Sean Carlisto de. **Análise de Alertas de Intrusão Baseada em Mineração de Processos**. 2016. 99f. Dissertação de Mestrado (Mestrado em Ciência da Computação) — Universidade Estadual de Londrina, Londrina, 2016.

RESUMO

Sistemas de Detecção de Intrusão (IDS - Intrusion Detection Systems) são dispositivos extensivamente utilizados como uma das camadas de proteção de uma rede a fim de evitar e mitigar as consequências causadas por violações de segurança. IDSs fornecem informações sobre as atividades intrusivas por meio de alertas que são avaliados manualmente por especialistas e auxiliam na decisão de um plano de resposta contra a intrusão. No entanto, uma das desvantagens dos IDSs está relacionada a grande quantidade de alertas gerados pelos dispositivos que faz com que a investigação manual se torne uma atividade onerosa e propensa a erros. Neste trabalho, é proposta uma abordagem para facilitar a investigação manual de grandes quantidades de alertas de intrusão por um especialista. A abordagem faz uso de técnicas de mineração de processos para extrair informações sobre o comportamento dos atacantes nos alertas e descobrir as estratégias de ataque multiestágio utilizadas por eles com a intenção de comprometer a rede. As estratégias são apresentadas para o administrador da rede em modelos visuais de alto nível e técnicas de clusterização hierárquica são aplicadas em modelos complexos de difícil compreensão com o objetivo de torná-los mais simples e intuitivos. Um conjunto de dados reais de alertas provenientes da Universidade de Maryland foram utilizados em um estudo de caso para avaliação e validação da abordagem proposta. A abordagem proposta combina características visuais com medidas quantitativas que auxiliam o administrador da rede na análise dos alertas de maneira mais compreensível se comparada a análise manual.

Palavras-chave: Detecção de intrusão. Visualização de segurança. Mineração de alertas.

ALVARENGA, Sean Carlisto de. **Analysis of Intrusion Alerts Based on Process Mining.** 2016. 99p. Dissertation (Master in Science in Computer Science) — State University of Londrina, Londrina, 2016.

ABSTRACT

Intrusion Detection Systems (IDS) are extensively used as one of the lines of defense of a network to prevent and mitigate the consequences caused by security breaches. IDS provide information about the intrusive activities on a network through alerts that are manually evaluated by specialists and help in determining a response plan against the intrusion. However, one of the downsides of IDS is the large amount of alerts triggered by them which makes the manual investigation of the alerts become a burdensome and error-prone activity. In this work, it is proposed an approach to facilitate the investigation of huge amounts of intrusion alerts by a specialist. The approach makes use of process mining techniques to extract information from the alerts regarding the attackers' behavior and discover the multistage attack strategies used by them to compromise the networks. Then, the strategies are presented to the network administrator in friendly high-level visual models and hierarchical clustering techniques are applied in complex models that are difficult to understand to make them more simple and intuitive. A real dataset of alerts from the University of Maryland was used in a case study for evaluation and validation of the proposed approach. The proposed approach combines visual features along with quantitative measures that help the network administrator to analyze the alerts in an easy and intuitive manner compared to manual analysis.

Keywords: Intrusion detection. Security visualization. Alert mining.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASN	*Autonomous System Number*
ASN.1	*Abstract Syntax Notation One*
BPM	*Business Process Modeling*
CERT/CC	*Computer Emergency Response Team Coordination Center*
CGI	*Common Gateway Interface*
CIA	*Confidentiality, Integrity and Availability*
CSV	*Comma-Separated Values*
CVE	*Common Vulnerabilities and Exposures*
DARPA	*Defense Advanced Research Projects Agency*
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
DDoS	*Distributed Denial of Service*
DNS	*Domain Name System*
EM	*Expectation Maximization*
FTP	*File Transfer Protocol*
HIDS	*Host-based Intrusion Detection System*
IDS	*Intrusion Detection Systems*
IP	*Internet Protocol*
IPS	*Intrusion Prevention System*
MulVAL	*Multi-host, Multi-stage, Vulnerability Analysis Language*
NBA	*Network Behavior Analysis*
NIDS	*Network-based Intrusion Detection System*
NVD	*National Vulnerability Database*
NetSPA	*a Network Security Planning Architecture*
RPC	*Remote Procedure Call*

SMB	*Server Message Block*
SO	Sistema Operacional
SOM	*Self-Organizing Maps*
SSH	*Security Shell*
SVM	*Support Vector Machine*
TCP	*Transmission Control Protocol*
TVA	*Topological Vulnerability Analysis*
WIDS	*Wireless Intrusion Detection System*
YaBB	*Yet another Bulletin Board*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o aumento na quantidade de novas vulnerabilidades de segurança descobertas tem preocupado as empresas e organizações. Somente no ano de 2013, mais de 5000 novas vulnerabilidades foram encontradas em programas de computadores e sistemas operacionais. Em 2014, o número de novas vulnerabilidades cresceu para aproximadamente 8000, um aumento de mais de 50% em relação ao ano anterior. Em 2015, foram descobertas cerca de 6500 novas vulnerabilidades. Os dados são de estatísticas apontadas pela Base de Dados Nacional de Vulnerabilidades (NVD - *National Vulnerability Database*) [4], que mostram ainda que nos últimos quatro anos, quase todas as vulnerabilidades reportadas são classificadas com severidade média e alta.

O aumento na quantidade de novas vulnerabilidades pode trazer consequências à segurança das redes de computadores e sistemas de informação. Quanto maior o número de novas vulnerabilidades, maior é a probabilidade de aumento no número de ocorrências de incidentes de segurança. A prevenção e o tratamento dos incidentes de segurança são dois dos grandes desafios enfrentados pelas organizações visto que os incidentes são responsáveis por causar inúmeros impactos negativos em diferentes áreas das organizações. Danos financeiros, gastos com tempo e recurso para recuperação do sistema comprometido e danos a reputação da organização em caso de roubo de informações confidenciais são apenas alguns exemplos dos prejuízos causados. Portanto, a aplicação de medidas protetivas para mitigar as consequências causados pelos incidentes torna-se uma tarefa imprescindível.

Sistemas de Detecção de Intrusão (IDS - *Intrusion Detection Systems*) são dispositivos que desempenham um papel importante na proteção das redes de computadores e sistemas de informação. IDSs monitoram a rede e as atividades dos sistemas para detectar quaisquer violações de segurança. Quando detecta-se uma violação de segurança, o evento intrusivo é reportado na forma de um alerta para o administrador da rede, que avalia a ameaça e inicia um plano de resposta contra a intrusão [5]. Infelizmente, IDSs podem disparar grandes quantidades de alertas que tornam a análise e identificação de alertas relevantes uma atividade impraticável [6]. Um exemplo desse cenário ocorre na Universidade de *Maryland*, nos Estados Unidos. Somente no ano de 2012, mais de 62 milhões de alertas foram disparados. Entre os alertas, quase 40 milhões de alertas foram registrados apenas no mês de novembro e mais de 35 milhões de alertas foram registrados somente no dia 06/11/2012. Esse cenário mostra que mesmo se fosse possível realizar a análise manual de um alerta a cada segundo, seria necessário mais de um ano para que o administrador da rede fosse capaz de avaliar todos os alertas. Além disso, por meio da análise manual, extrair informações relevantes que ajudam a entender e interpretar os eventos intrusivos

de uma quantidade grande de alertas torna-se uma tarefa complexa.

Para abordar esse problema, diversas técnicas têm sido propostas para auxiliar o administrador da rede na análise e extração de informações relevantes de grandes volumes de alertas. As técnicas propostas, de modo geral, podem ser classificadas em duas categorias: técnicas de preprocessamento de alertas e técnicas de correlação de alertas [7]. As abordagens de preprocessamento de alertas defendem que grande parte dos alertas disparados por um IDS são falsos positivos gerados a partir de uma mesma causa, chamada de causa raiz. Segundo essa abordagem, as causas raiz são responsáveis por gerar um número grande de alertas redundantes que aumentam a quantidade de alertas disparados por um IDS. Desse modo, as abordagens de preprocessamento têm como objetivo reduzir o número de falsos positivos, identificando e corrigindo as causas raiz dos alertas. Por outro lado, as abordagens de correlação de alertas defendem que as informações contidas em grandes quantidades de alertas podem ser generalizadas e representadas por um número menor de alertas. Dessa forma, a abordagem busca agrupar os alertas, de modo que novos significados e informações de alto nível possam ser inferidas a partir dos grupos formados.

Alguns trabalhos [6, 8, 9] que utilizam técnicas de correlação de alertas têm sido propostos para extrair descritores de alto nível de grandes volumes de alertas. As informação extraídas são representadas em modelos gráficos que descrevem as estratégias e cenários dos ataque que ocorrem em uma rede. Entretanto, a ideia de usar descritores de alto nível e modelos gráficos para avaliação de segurança não é exclusiva de pesquisas de correlação de alertas, mas é também empregada nas teorias de árvores de ataque e grafos de ataque. Árvores de ataque e grafos de ataque têm sido usados extensivamente para uma variedade de propósitos, como na modelagem de ameaças de segurança para identificação de vulnerabilidades do sistema e na quantificação de métricas de segurança, por exemplo, para estimar a probabilidade de ocorrência de um ataque [10]. No entanto, essas representações geralmente exigem a intervenção de um especialista para gerar o modelo, já que necessitam de informações sobre diferentes aspectos da rede como a topologia utilizada, número de *hosts* e configurações do *firewall* [10].

Neste trabalho, é proposta uma abordagem de correlação de alertas com ênfase em modelos visuais para auxiliar o administrador da rede na investigação de grandes volumes de alertas que são gerados pelos IDS. A abordagem tem como objetivo minerar os alertas e representar as informações referentes aos ataques em modelos de alto nível. Os modelos são representações visuais que descrevem as estratégias utilizadas pelos atacantes nas tentativas de comprometer a rede, extraídas a partir do seu comportamento observado nos alertas. O principal foco da abordagem são as estratégias de ataque multiestágio, nos quais os atacantes executam uma sequência de etapas a fim de atingir seu objetivo. A abordagem é dividida em quatro etapas. Primeiramente, considerando uma perspectiva em relação aos ataques, os alertas são organizados em grupos que podem estar relaci-

onados a uma mesma estratégia de ataque executada contra à rede. Em seguida, uma seleção de atributos é realizada nos grupos de alertas formados, que são posteriormente convertidos em um formato adequado de log de eventos para a mineração de processos. A partir do comportamento dos atacantes observado no log de eventos, as estratégias de ataque multiestágio utilizadas pelos atacantes são obtidas e representadas por meio de um modelo visual de alto nível. O modelo é gerado por um algoritmo baseado em técnicas de descoberta de modelos da mineração de processos e é apresentado para o administrador da rede para análise.

Um dos principais objetivos da abordagem proposta é sintetizar grandes quantidades de alertas de IDS em modelos visuais de fácil compreensão. No entanto, em algumas situações, os modelos podem se tornar grandes e complexos, o que dificulta sua análise. Portanto, na última etapa da abordagem, a redução de complexidade dos modelos gerados é realizada utilizando técnicas de clusterização hierárquica. A clusterização dos modelos é efetuada por meio de um processo de discretização dos grupos de alertas, em que grupos que apresentam comportamento similares são reorganizados em grupos homogêneos utilizando a clusterização hierárquica com o método de *Ward* e a distância de Jaccard [11, 12].

Para avaliar a abordagem proposta, um estudo de caso foi realizado utilizando um conjunto de dados de alertas da Universidade de *Maryland*. No estudo de caso, buscou-se avaliar a abordagem que tem como objetivo alcançar as seguintes contribuições: i) representar informações complexas como no caso de grandes volumes de alertas de IDS em modelos visuais intuitivos e compreensíveis, ii) apresentar as estratégias de ataque multiestágio que estão sendo utilizadas contra a rede para auxiliar os administradores na identificação de serviços que são alvos dos ataques e possibilitar que medidas preventivas e correções de segurança sejam acionadas e iii) apresentar por meio dos modelos, informações quantitativas sobre a frequências dos ataques para auxiliar o administrador a priorizar as ações protetivas contra eles. O restante do trabalho está organizado como segue:

- O Capítulo 2 apresenta a fundamentação teórica, definindo os principais conceitos sobre IDS, árvores de ataque e grafos de ataque. Em seguida, é apresentado um levantamento de trabalhos de análise de alertas de IDS com o objetivo de contextualizar a abordagem proposta dentro da literatura existente. Por fim, são apresentados os conceitos sobre clusterização hierárquica e mineração de processos que são utilizados na abordagem proposta.

- O Capítulo 3 apresenta as quatro etapas da abordagem proposta: agregação dos alertas de IDS, conversão dos alertas agregados em um log de eventos, descoberta do modelo de ataque e clusterização automática dos modelos complexos.

- O Capítulo 4 apresenta a avaliação e validação da abordagem proposta, apresentando os resultados obtidos em um estudo de caso realizado a partir de conjunto de dados reais provenientes de alertas disparados na rede da Universidade de *Maryland*.
- O Capítulo 5 apresenta a conclusão, discutindo as considerações finais e as direções para trabalhos futuros.

2 PRINCIPAIS DEFINIÇÕES

Este capítulo apresenta uma visão geral dos conceitos relacionados à sistemas de detecção de intrusão, análise de dados de segurança de rede e os principais conceitos utilizados no decorrer do trabalho. Primeiramente, são apresentados os conceitos de intrusão e métodos automatizados que realizam o processo de detecção de intrusão. Logo depois, são apresentados métodos que utilizam representações visuais de alto nível para auxiliar o administrador na análise e avaliação da segurança da rede. Em seguida, é apresentado um levantamento de trabalhos que propõem abordagens para auxiliar o administrador da rede na análise de dados de segurança, especificamente de alertas de IDS. Por fim, são apresentados os conceitos sobre clusterização hierárquica e mineração de processos que são utilizados pela abordagem proposta.

2.1 Sistemas de Detecção de Intrusão

Intrusões são eventos de segurança provenientes de ações praticadas por agentes maliciosos, isto é, intrusos, que ameaçam as propriedades fundamentais da segurança da informação [13]. Segurança da informação é um conceito que se refere à proteção e preservação contra o acesso não autorizado, uso, divulgação não autorizada, interrupção, modificação ou destruição da informação e sistemas da informação [14]. Segundo a definição dada pela norma ISO/IEC 27000:2014, "segurança da informação é a preservação da confidencialidade, integridade e disponibilidade da informação" [15]. Confidencialidade, integridade e disponibilidade, conhecidas como tríade CIA (*Confidentiality, Integrity and Availability*), são propriedades que formam os objetivos e princípios da segurança da informação [14, 16, 17]:

- **Confidencialidade:** propriedade que protege a informação contra acessos não autorizados.
- **Integridade:** propriedade que previne a alteração não autorizada ou indesejada da informação.
- **Disponibilidade:** propriedade que garante a acessibilidade e uso da informação quando requisitada por agentes autorizados.

Uma intrusão pode ser classificada como interna ou externa. Uma intrusão interna é causada por um agente com acesso autorizado à rede alvo, que busca aumentar seu nível de privilégios para acessar informações às quais não teria acesso com seu nível inicial, como acessar dados sensíveis e sigilosos de uma organização. Uma intrusão externa é causada

por um agente sem acesso à rede alvo, que busca ganhar acesso às informações do sistema explorando suas falhas e brechas de segurança. A automatização da detecção de intrusões, sejam elas internas ou externas, pode ser realizada por meio de IDSs [16, 17, 18].

O IDS é um programa ou equipamento que monitora computadores ou tráfego de rede procurando por violações de segurança. Uma vez que a atividade maliciosa é detectada, o IDS pode responder de diferentes formas, como gerar um alerta para notificar o administrador da rede ou registrar o evento ocorrido [17]. Após notificado sobre o evento, o administrador da rede avalia a ameaça e dá início a um plano de resposta contra a intrusão.

IDSs geram alertas para quaisquer eventos suspeitos ocorridos na rede. Entretanto, nem todos eventos suspeitos detectados são provenientes de uma intrusão. Alertas dessa natureza são chamados de falsos positivos. Em contrapartida, atividades intrusivas não detectadas pelo dispositivo, isto é, sem a ocorrência de alertas, são chamadas de falsos negativos. A redução de alertas falsos positivos é uma área que tem recebido bastante atenção na literatura. Uma descrição completa de métodos e técnicas que abordam o problema, podem ser encontradas na revisão bibliográfica apresentada por [19].

IDSs podem ser classificados como sistemas passivos ou ativos. Um sistema passivo monitora e identifica quaisquer ações suspeitas que indiquem uma possível violação de segurança que visa comprometer a rede. Nenhuma medida pode ser tomada contra a ameaça. Por outro lado, um sistema ativo, além de funções de monitoramento, pode responder às atividades suspeitas encerrando a conexão, descartando os pacotes maliciosos e/ou bloqueando a comunicação com a fonte suspeita. Essa segunda categoria de IDS é conhecida como IPS (*Intrusion Prevention System*) [16, 18].

Um dos aspectos importantes dos IDSs é o local de sua implantação na rede [18]. Com relação a esse aspecto, os IDSs podem ser classificados em duas principais categorias: IDS baseado em rede e IDS baseado em *host*. A descrição de outras categorias como *Wireless* IDS (WIDS), além de outros métodos de detecção como sistemas NBA (*Network Behavior Analysis*) são apresentados em [20].

2.1.1 IDS baseado em rede

Um NIDS (*Network-based Intrusion Detection System*) é implantado em determinados segmentos da rede e monitora as atividades desse segmento procurando por ações maliciosas. NIDSs são comumente implantados nos limites entre duas redes, por exemplo, na proximidade de roteadores de borda, com o propósito de monitorar todo tráfego de entrada e saída da rede [20]. Entretanto, outras possibilidades de implantação podem ser adotadas. Por exemplo, para detecção de ataques que conseguem passar pelo *firewall*, o NIDS pode ser implantado entre a rede protegida e o *firewall* externo que faz fronteira com a Internet. Outra opção é implantá-lo entre o *firewall* externo e a Internet, o que permite

coletar informações como o número de ataques provenientes da Internet direcionados à rede [18].

NIDSs podem ser implantados em um dos dois modos: modo *inline* e modo passivo. No modo *inline*, o NIDS é implantado na rede de modo que todo o tráfego de rede deve passar pelo dispositivo [18, 20]. Dessa maneira, o modo *inline* oferece medidas preventivas, visto que o tráfego pode ser bloqueado (no caso de um IDS ativo) antes de chegar à rede. No entanto, esse modo pode provocar um gargalo no tráfego de rede, contribuindo para o atraso na entrega de pacotes [18]. Em vista disso, no modo passivo, apenas uma cópia do tráfego passa pelo NIDS. Isto pode ser realizado por meio de um *tap* de rede, que conecta o NIDS com o meio físico (por exemplo, por meio de um cabo de fibra óptica) ou espelhamento de porta em *switch* [18, 20].

2.1.2 IDS baseado em *host*

Um HIDS (*Host-based Intrusion Detection System*) é implantado em um sistema e monitora e analisa não somente o tráfego de rede, mas também chamadas de sistema, modificações em arquivos, comunicação entre processos, registros de aplicações, etc. [16]. Alguns HIDSs, conhecidos como *application-based* IDS, são projetados para monitorar atividades e serviços de uma aplicação específica, por exemplo, um servidor Web. Portanto, HIDSs são comumente implantados em *hosts* críticos como servidores contendo informações sensíveis e servidores com acesso à Internet [20]. HIDSs possuem algumas limitações, como o consumo considerável de recursos computacionais dos *hosts* sob proteção, além da possibilidade de gerar conflitos com outros controles de segurança, por exemplo, *firewalls* pessoais [20].

2.1.3 Métodos de detecção

Em relação aos métodos de detecção, IDSs podem utilizar diferentes técnicas como detecção baseada em assinaturas, detecção baseada em anomalias e detecção híbrida.

A detecção baseada em assinaturas é o processo de comparar padrões ou assinaturas que correspondem a uma ameaça conhecida a eventos observados na rede para identificar atividades maliciosas. Essa técnica utiliza uma base de dados de assinaturas de ataques já conhecidos para detectar a intrusão. IDSs baseados em assinaturas são muito eficazes para detecção de ataques conhecidos, já definidos em sua base de dados. Por outro lado, são incapazes de detectar ataques que não possuem assinaturas definidas como ataques do tipo *zero-day* ou ataques modificados [16]. Essa limitação pode ser contornada adicionando novas assinaturas na base de dados do dispositivo e mantendo a base de dados atualizada.

IDSs que utilizam a detecção por assinatura também são conhecidos como IDSs baseados em conhecimento, uma vez que exigem uma base de conhecimento/dados predefi-

nida para realizar a análise e detecção [21]. Sistemas especialistas, por exemplo, classificação baseada em regras, máquinas de estados finitos e linguagens descritivas, são exemplos de técnicas empregadas em abordagens de detecção por assinatura [21, 22, 23]. A seguir, um exemplo de uma assinatura baseada em regras utilizada pelo IDS *open source* Snort [24] é apresentada.

```
alert tcp EXTERNAL_NET -> HOME_NET
    (flags: FPU; msg: "Possível port scan detectado!";)
```

Nessa regra, um alerta é gerado caso seja detectado um pacote TCP (*Transmission Control Protocol*) vindo da rede externa, endereçado para a rede interna e com as *flags* do cabeçalho TCP FIN, PSH e URG assinaladas. Pacotes TCP dessa natureza podem indicar uma possível tentativa de *port scan* e, portanto, são detectados pelo IDS.

Um IDS baseado em anomalias, por outro lado, opera distinguindo um comportamento anormal do que é considerado normal. Dessa maneira, essa técnica constrói um modelo de tráfego de rede normal e gera um alerta para qualquer tráfego que se desvie desse modelo. Uma grande vantagem do método é a detecção de novos ataques sem necessitar de qualquer conhecimento prévio sobre eles [20]. Uma limitação do método de detecção baseado em anomalias é a dificuldade em se definir um modelo para o que é considerado normal, o que é malicioso e definir qual o desvio significativo entre os modelos para que seja disparado o alerta [16].

IDSs que utilizam a detecção por anomalias também são conhecidos como IDSs baseados em comportamento. Métodos estatísticos, por exemplo, métodos para análise de séries temporais, e abordagens de aprendizado de máquina como modelos de Markov, redes Bayesianas, Lógica *Fuzzy* e Algoritmos Genéticos, são alguns exemplos de técnicas empregadas em abordagens de detecção por anomalias [21, 22, 23].

Um método híbrido combina as vantagens dos métodos baseados em assinatura e em anomalias e os integra em um único sistema.

2.2 Análise de dados de segurança

Nesta seção, são apresentados os conceitos relacionados à visualização da segurança da rede e análise de dados de segurança. A Seção 2.2.1 apresenta as árvores de ataque e grafos de ataque, duas representações visuais de alto nível utilizadas na modelagem de ameaças de segurança. Em seguida, a Seção 2.2.2 apresenta abordagens que auxiliam a análise de alertas de IDS encontradas em trabalhos presentes na literatura.

2.2.1 Grafos e árvores de ataque

As árvores de ataque foram introduzidas por Schneier [25, 26] e denotam uma representação visual, que tem por objetivo realizar a modelagem de um ataque em uma notação estruturada em árvore. Árvores de ataque fornecem uma maneira sistemática de análise e modelagem de ameaças de segurança, apresentando de forma hierárquica as diferentes maneiras em que um sistema computacional pode ser comprometido.

Em uma árvore de ataque, os nós representam um objetivo a ser alcançado. A meta do atacante, isto é, o objetivo principal, é especificada como a raiz da árvore. Os ramos da árvore representam as submetas ou condições que devem ser satisfeitas pelo atacante a fim de alcançar determinado objetivo. Os nós podem ser representados como nós disjuntivos (alternativas) ou conjuntivos (agregação) [26]. Nós disjuntivos descrevem os caminhos alternativos que um atacante pode seguir para alcançar seu objetivo. Nós conjuntivos representam as diferentes etapas que um atacante precisa tomar para alcançar um objetivo. Ramos descendentes a partir da raiz refinam o ataque em novas submetas até que não seja mais possível realizar o refinamento (nós folhas). A Figura 2.1 mostra um exemplo de uma árvore de ataque.

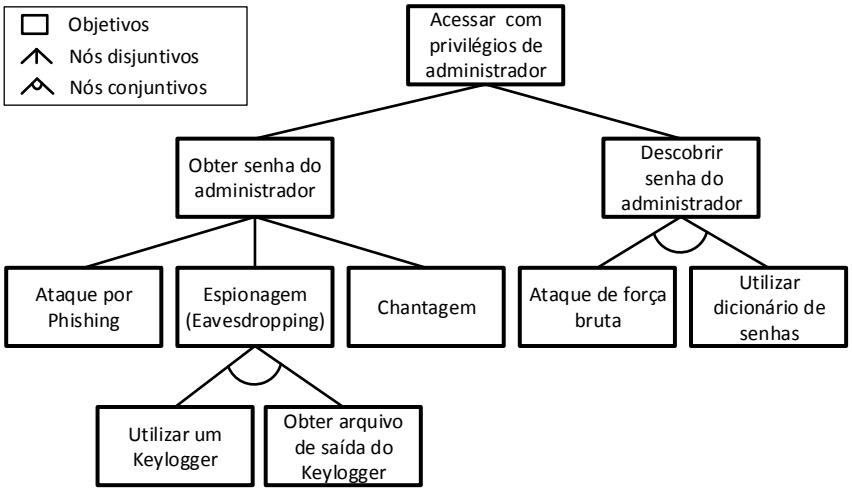


Figura 2.1 – Árvore de ataque para acessar um sistema com nível de privilégios de administrador.

Na árvore de ataque da Figura 2.1, o objetivo do atacante é acessar um sistema com nível de privilégios de administrador. Para alcançar esse objetivo, há duas possibilidades para o atacante: obter a senha do administrador ou descobrir a senha do administrador. Essas opções são representadas por dois ramos da árvore ligados à raiz. Para obter a senha do administrador, o atacante tem três alternativas: i) obter por meio de *phishing*, uma tentativa de fraude eletrônica que tem como objetivo obter dados sensíveis da vítima por meio de comunicação eletrônica (*emails* ou *Web sites*) em que o atacante se passa por uma entidade confiável [16], ii) por meio de espionagem ou iii) por meio de chantagem. Para espionar a pessoa responsável pela senha, o atacante pode utilizar um *sniffer* de teclado

(*keylogger*) e, posteriormente, reaver o arquivo de saída do programa/dispositivo. Ambos os requisitos devem ser satisfeitos para o sucesso do objetivo do atacante. Por outro lado, para descobrir a senha do administrador, o atacante pode realizar um ataque de força bruta em conjunto com um dicionário de senhas frequentemente utilizadas.

As aplicações das árvores de ataque vão além da modelagem gráfica e sistemática dos ataques. A partir do modelo, a avaliação de ameaças de segurança, como determinar a viabilidade de um ataque, probabilidade de sucesso, dificuldade na execução, além da quantificação de métricas, como o custo de um ataque, podem ser realizadas [26]. A Figura 2.2 apresenta um exemplo de árvore de ataque para determinar a viabilidade de um ataque.

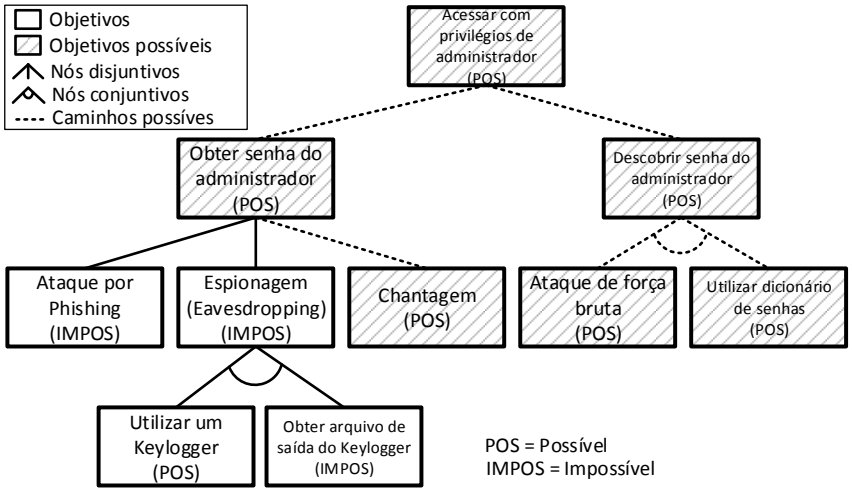


Figura 2.2 – Árvore de ataque para determinar a viabilidade de um ataque.

Para modelar a árvore de ataque da Figura 2.2, primeiramente atribui-se aos nós folhas valores referentes à possibilidade (POS) ou não (IMPOS) do objetivo ser alcançado pelo atacante. Em seguida, os valores são propagados para os outros nós da árvore (ramos) até chegar à raiz. Aos nós disjuntivos é atribuído o valor POS, se pelo menos uma de suas condições é satisfeita. Em contrapartida, atribui-se o valor POS a nós conjuntivos somente se todas as suas condições foram satisfeitas. Em todos outros casos atribui-se IMPOS. Portanto, para acessar um sistema com nível de privilégios de administrador, a senha do administrador deve ser obtida por meio de chantagem ou descoberta por meio de um ataque de força bruta. As alternativas para o ataque são destacadas na árvore por linhas tracejadas, e as etapas do ataque por áreas hachuradas. Com base nessas informações, o administrador do sistema pode tomar medidas preventivas para reforçar os pontos vulneráveis do sistema e ter conhecimento de como proteger o sistema contra o ataque.

Árvores de ataque são frequentemente modeladas de forma manual. O processo envolve informações que podem abranger diversos aspectos da rede e variar de acordo com o que deseja-se modelar. Por exemplo, para determinar a quais ataques um sistema

é vulnerável, uma investigação meticulosa sobre os pontos fracos do sistema deve ser realizada. A investigação deve considerar informações sobre a topologia da rede, a existência e a localização de um *firewall*, de um IDS, etc. Além disso, algumas modelagens podem necessitar de informações que não são possíveis de obter. Por exemplo, para determinar o sucesso de um ataque, pode ser necessário conhecer as características do atacante para determinar se ele possui ou não as habilidades necessárias para sua execução. Dessa maneira, o processo de modelagem se torna trabalhoso, propenso a erros e que pode não refletir o estado real da rede, podendo levar a conclusões equivocadas.

A abordagem proposta neste trabalho tem foco nos ataques. Nesse sentido, a abordagem utiliza alertas de IDS para identificar os comportamentos e estratégias que atacantes estão empregando para tentar comprometer a rede. No entanto, diferentemente da modelagem de árvores de ataque, nenhuma suposição é feita acerca de informações sobre possíveis vulnerabilidades e habilidades dos atacantes. Consequentemente, o modelo construído pela abordagem representa o que de fato ocorreu e o que está acontecendo, considerando apenas violações de segurança realizadas pelas ações dos atacantes que são reportadas nos alertas. Além disso, árvores de ataque possuem algumas limitações relacionadas a modelagem de ataques. A representação é considerada estática já que não leva em consideração aspectos temporais, como variação do tempo e ordem ou prioridade das ações [10]. Portanto, esse modelo de representação não é adequado para a abordagem em que a ordem das ações dos atacantes é um aspecto determinante, uma vez que ela denota o comportamento dos atacantes que é representado nos modelos visuais gerados.

Os grafos de ataque, termo introduzido pela primeira vez por Phillips e Swiler [27, 28], são outra maneira de representar e analisar ataques. Um grafo de ataque é um formalismo usado por analistas de segurança para modelar as vulnerabilidades de segurança de uma rede. Em um grafo de ataque, é possível representar o relacionamento entre diferentes componentes da rede (*hosts*, *firewall*, seviços, IDS, etc.) permitindo ponderar as consequências de possíveis ataques em determinados contextos e situações [29]. As aplicações do formalismo incluem diversas áreas como análise da segurança da rede (vulnerabilidades), na detecção de intrusão (em conjunto com IDS para detecção de cenários de ataque conhecidos), estratégias de defesas (medidas preventivas, *hardening* da rede) e análise forense [30].

Grafos de ataque descrevem os caminhos por meio dos quais um intruso pode explorar as vulnerabilidades de um sistema para atingir um objetivo. Os nós do grafo são descritos na forma *vulnerabilidade(fonte, alvo)* e representam o estado da rede. As arestas representam uma ação do atacante que altera o estado da rede, descritos na forma *condição(fonte)*, para condição envolvendo um único *host*, ou *condição(fonte, alvo)*, para condições envolvendo mais de um *host*. Pesos podem ser atribuídos nas arestas para aprimorar o modelo e algoritmos em grafos podem ser aplicados para encontrar o caminho

mais provável de sucesso, o tempo de sucesso, entre outras métricas [10]. A Figura 2.3 mostra um exemplo de grafo de ataque.

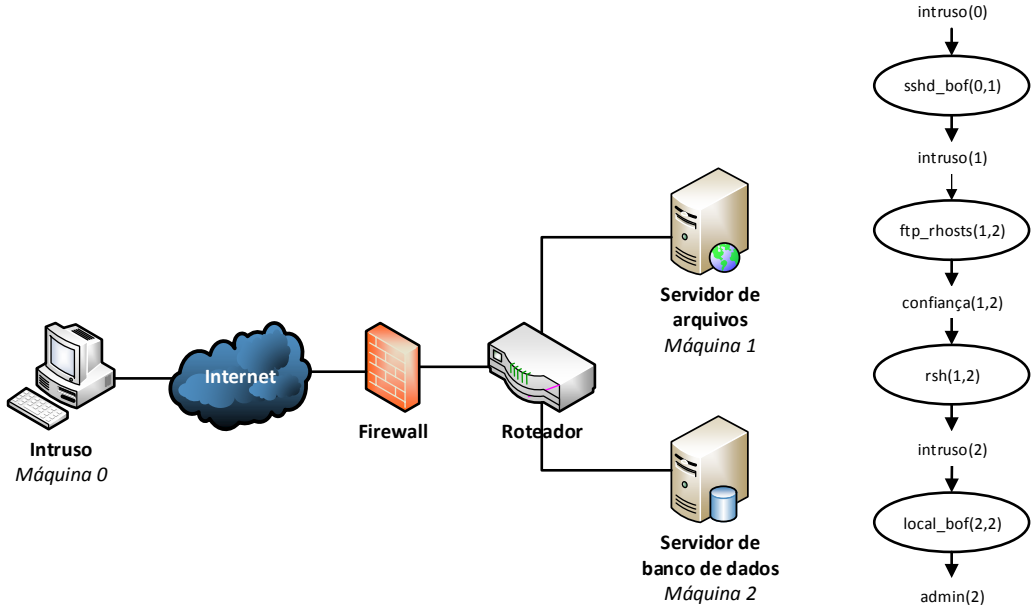


Figura 2.3 – Exemplo de uma rede e um possível grafo de ataque. Adaptado de [1].

O grafo de ataque da Figura 2.3 apresenta apenas um dos possíveis caminhos de ataque para obter privilégios de administrador no servidor de banco de dados localizado na rede. O caminho do ataque inicia-se com *sshd_bof(0,1)*. Um intruso (Máquina 0) explora via SSH (*Security Shell*) a vulnerabilidade do tipo *Buffer Overflow* contra o servidor de arquivos (Máquina 1). A vulnerabilidade permite que o intruso execute códigos no servidor de arquivos como um usuário legítimo. Com o controle do servidor de arquivos (condição *intruso(1)*), o intruso então explora uma vulnerabilidade no serviço FTP (*File Transfer Protocol*) contra o servidor de banco de dados (Máquina 2) para realizar anonimamente um *upload* de uma lista de *hosts* confiáveis. Estabelecida a relação de confiança entre os servidores (*confiança(1,2)*), agora é possível a execução remota de comandos *shell* no servidor de banco de dados sem a necessidade de autenticação com senha (*rsh(1,2)*). Após controlar o servidor do banco de dados (*intruso(2)*), o intruso pode explorar a vulnerabilidade do tipo *Buffer Overflow* local no servidor (*local_bof(2,2)*) e executar códigos no servidor com todos privilégios (*admin(2)*).

Diversas abordagens que propõem métodos automatizados para geração de grafos de ataque são encontradas na literatura como TVA (*Topological Vulnerability Analysis*), NetSPA (*a Network Security Planning Architecture*) e MulVAL (*Multi-host, Multi-stage, Vulnerability Analysis Language*) [31]. Entretanto, os grafos de ataque sofrem das mesmas limitações das árvores de ataque uma vez que requisitam diferentes tipos de informações (vulnerabilidades do sistema, topologia da rede, perfil do atacante, etc. [10]) que dependem do conhecimento de um especialista. Logo, são propensas a erros. Além disso, algumas

informações podem não ser conhecidas o que reforça um dos benefícios da abordagem propostas neste trabalho, que utiliza apenas alertas de IDS.

2.2.2 Análise de alertas de intrusão

Nesta seção, são apresentados diversos trabalhos que abordam o problema do grande volume de alertas gerados por IDSs e propõem soluções que auxiliam o administrador da rede a realizar a análise desses alertas. De modo geral, as técnicas apresentadas podem ser classificadas em duas categorias: técnicas de preprocessamento de alertas e técnicas de correlação de alertas [7]. As técnicas de preprocessamento de alertas buscam reduzir a influência de falsos positivos nos conjuntos de alertas. Sendo assim, técnicas de preprocessamento de alertas são frequentemente utilizadas em etapas iniciais das abordagens com o objetivo de melhorar a precisão dos resultados das próximas etapas. As técnicas de correlação de alertas têm como objetivo analisar grandes volumes de alertas e fornecer informações relevantes que deem significado e ajudem na interpretação e investigação desses alertas. Na correlação de alertas, as informações são frequentemente sintetizadas e apresentadas para o administrador da rede como alertas de alto nível chamados de hiper-alertas ou meta-alertas [7]. Em alguns trabalhos de correlação, representações visuais de alto nível são utilizadas para descrever os cenários e estratégias de ataque que os atacantes estão utilizando, visando comprometer a rede. A seguir, os trabalhos relacionados serão apresentados em ordem cronológica.

Dois dos primeiros trabalhos a abordar o problema do grande volume de alertas gerados por IDSs foram realizados por Julisch e Dacier [32, 33]. Em seus trabalhos, os autores propõem um método de redução de volume de alertas por meio da análise da causa raiz. A causa raiz de um alerta é definida como o motivo pelo qual o alerta foi disparado. Julisch e Dacier observaram que um pequeno número de causas raiz é responsável por mais de 90% da ocorrência dos alertas. A partir dessa observação, os autores propõem uma abordagem que visa identificar e solucionar a causa raiz dos alertas. Por exemplo, um erro de configuração em um servidor pode ser a causa da ocorrência de diversos alertas. A correção do erro remove a causa raiz e, portanto, reduz o volume de alertas disparados no futuro.

Em geral, técnicas de correlação de alertas têm como objetivo reduzir o volume de alertas que devem ser investigados por um especialista. Entretanto, algumas abordagens utilizam a correlação de alertas para extrair informações relevantes dos grandes volumes de alertas. Em dois trabalhos publicados por Ning *et al.* [6, 34], é proposto um método de correlação para descobrir estratégias de ataques a partir de alertas de IDS. A técnica proposta parte do princípio de que a grande maioria dos ataques está organizado em etapas, em que ataques de etapas anteriores preparam os ataques de etapas posteriores. Assim, a técnica proposta correlaciona os alertas utilizando conceitos de pré-requisitos

de intrusão (condição necessária para o ataque ter sucesso) e consequência de intrusão (o resultado do ataque caso tenha sucesso). Os alertas correlacionados são representados na forma de hiper-alertas e são apresentados visualmente em um grafo. Em seu trabalho mais recente, Ning e Xu [6] aprimoram a abordagem anterior, desenvolvendo um método para medir a similaridade entre diferentes grafos de estratégias de ataque.

Em Treinen e Thurimella [35], a correlação de alertas é realizada utilizando técnicas de mineração de dados. Os autores propõem um *framework* que utiliza regras de associação para encontrar o relacionamento e dependências causais entre os alertas. Regras de associação são algoritmos de mineração de dados que têm como objetivo encontrar um relacionamento entre membros pertencentes a grandes conjuntos de dados. Na abordagem proposta, os alertas são modelados em um grafo direcionado, em que os vértices representam os endereços IP de origem ou destino do alerta e as arestas conectam os vértices da origem para o destino. Para cada componente conectado, regras de associação são derivadas para auxiliar o administrador da rede a criar novos padrões para identificar os ataques.

Em Lee *et al.* [36], é proposto um método de correlação de alertas por meio de cálculos de similaridade entre os atributos dos alertas. Os autores desenvolveram um sistema que filtra alertas redundantes e os agrega em hiper-alertas com base no endereço IP de origem e na classe do ataque. Em seguida, a similaridade entre os hiper-alertas é calculada, levando em conta características como o endereço IP, as portas de origem e destino, a classe do ataque e o *timestamp* do alerta.

Assim como o trabalho de Treinen e Thurimella [35], regras de associação também são utilizadas em [37]. Os autores propõem uma abordagem que utiliza técnicas de clusterização e regras de associação para correlacionar alertas de um IDS. Na abordagem proposta, os alertas são preprocessados para selecionar os atributos que serão utilizados para análise como o endereço IP de origem e destino, portas de origem e destino, comprimento do pacote IP e o *timestamp*. A seguir, alertas com atributos semelhantes são agrupados utilizando o algoritmo de clusterização EM (*Expectation Maximization*). Na próxima etapa, para cada grupo de alertas formado, regras de associação são derivadas utilizando o algoritmo *Apriori*. Por fim, cada regra derivada é rotulada como sendo de ataques ou de tráfego de rede normal ou suspeito.

Abordagens que utilizam métodos da área da Inteligência Computacional para análise de alertas de IDS, também são encontrados na literatura, como o trabalho de Soleimani e Ghorbani [38]. Os autores propõem um método de filtragem dos alertas para identificação de alertas críticos e ataques multiestágio utilizando árvores de decisão. Primeiramente, os alertas de IDS são separados em episódios, isto é, em sequências de alertas ocorridos dentro de uma janela de tempo t, ordenados pelo atributo *timestamp*. Em seguida, episódios de comprimento um e episódios de comprimento maior que um são classi-

ficados como sendo críticos ou não. A classificação é realizada utilizando árvores de decisão com base no conhecimento de um especialista, que fornece informações sobre serviços e servidores críticos da rede. Em outro trabalho de sua coautoria, Sadoddin e Ghorbani [39] propõem o algoritmo *FSP_Growth*, que tem como objetivo minerar padrões frequentes dos alertas considerando sua estrutura. O algoritmo proposto é baseado no algoritmo de mineração de padrões frequentes *FP_Growth*. No trabalho proposto, grupos de alertas (chamados de padrões) são formados e mantidos em uma estrutura de dados em árvore chamada de árvore de padrões. A estrutura é mantida atualizada e padrões de ataque são minerados utilizando o novo algoritmo proposto.

Um diferencial em algumas abordagens é a ênfase em modelos visuais para análise dos alertas de intrusão, estratégia adotada nos trabalhos de Dasireddy *et al.* [40] e Yang *et al.* [41]. Os trabalhos propõem um modelo de visualização de alertas de IDS para auxiliar o administrador da rede na decisão de um plano de resposta contra as tentativas de intrusão. O modelo é composto por dois componentes. O primeiro componente processa alertas provenientes do IDS *Snort* e exibe um modelo gráfico da topologia da rede e suas conexões. O segundo componente realiza a clusterização dos alertas, em que Dasireddy *et al.* [40] utiliza a distância de *Hamming* e Yang *et al.* [41] utiliza n-gramas para o cálculo de similaridade entre os alertas. O processo de clusterização pode ser visualizado em um modelo que exibe a formação dos agrupamentos por alertas similares. Além disso, Yang *et al.* [41] aprimora sua abordagem, desenvolvendo um terceiro componente, responsável por detectar padrões nos alertas utilizando o algoritmo de regras de associação *Apriori*.

Em Liu *et al.* [42] é proposto um método de correlação de alertas com base em autômatos finitos. A abordagem utiliza autômatos finitos não-determinísticos para modelar cenários de ataques a partir de alertas de IDS. Os cenários de ataque representam três perspectivas do atacante em relação a rede. Os alertas são correlacionados considerando cinco classes. Cada classe representa os passos de um ataque como reconhecimento, atividades de invasão, elevação de privilégios, atividades de *host* e eliminação de vestígios. Um módulo de visualização é responsável por gerar um grafo para cada cenário de ataque construído, que posteriormente é apresentado ao administrador da rede.

Em [43], técnicas de clusterização são utilizadas para reduzir o volume de alertas gerados pelos IDSs. Os autores propõem um método para agrupar alertas semelhantes, em que cada grupo formado representa um hiper-alerta. A similaridade entre os alertas é calculada considerando os atributos de endereços IP, portas e assinatura do alerta, além de três novos atributos proposto na abordagem: relevância do alerta, severidade do alerta e frequência do alerta. Os novos atributos são calculados comparando os atributos dos alertas a bases de dados de vulnerabilidades e logs da rede e têm como objetivo enriquecer os alertas com informações adicionais para auxiliar o processo de clusterização de alertas semelhantes.

O trabalho de Taha *et al.* [44] propõe uma abordagem de correlação de alertas que utiliza agentes inteligentes para guiar o processo de correlação, propondo seis componentes. Cada componente se difere dos outros em relação aos critérios e atributos utilizados para agregar e correlacionar os alertas. A taxa de redução de alertas de cada componente está relacionada ao conjunto de dados de alertas utilizado. A abordagem faz uso de um agente inteligente para determinar quais componentes serão utilizados no processo de correlação, além da ordem em que serão aplicados. O agente aprende sobre os conjuntos de dados de alertas utilizando uma base de conhecimento previamente determinada. Seguindo a mesma linha, o trabalho de Cipriano *et al.* [45], propõe a ferramenta Nextat para predição de ataques. A ferramenta utiliza técnicas de aprendizado de máquina para aprender sobre o comportamento dos atacantes por meio do histórico de alertas de IDS, e posteriormente, utiliza esse conhecimento para predizer a probabilidade de novos ataques.

Saad e Traore [46] propõem um método de agregação de alertas baseado em similaridades semânticas entre os atributos dos alertas. A abordagem utiliza o conceito de ontologia e a taxonomia das classes de ataques para medir a similaridade entre dois alertas distintos. Posteriormente, os grupos de alertas formados são representados como hiper-alertas, reduzindo o grande volume de alertas gerados pelo dispositivo IDS.

Em Ahmadinejad *et al.* [47], é proposta uma abordagem híbrida em dois níveis de correlação de alertas. O primeiro nível correlaciona os alertas por meio de vulnerabilidades modeladas em um grafo de ataque. O segundo nível correlaciona os alertas por meio da similaridade entre seus atributos caso a vulnerabilidade não seja conhecida (definida no primeiro nível).

Lagzian *et al.* [8] propõem uma abordagem que utiliza técnicas da mineração de dados semelhante a utilizada por Sadoddin e Ghorbani [39]. No trabalho proposto, os autores utilizam uma estrutura em grafo para agrupar os alertas e, em seguida, aplicam o algoritmo *Bit-AssocRule* para minerar cenários de ataques frequentes do grafo e extrair as estratégias de ataque que são apresentadas para o administrador da rede.

Em [48] é proposta uma abordagem que tem como objetivo reduzir o volume de alertas disparados por dispositivos IDS, além de reduzir o volume de alertas falsos positivos. A abordagem é dividida em duas etapas. Na primeira etapa, os alertas são correlacionados em hiper-alertas com base na similaridade entre seus atributos, utilizando as técnicas de clusterização de Mapas Auto-Organizáveis (SOM - *Self-Organizing Maps*) e o algoritmo Neural GAS. Posteriormente, os hiper-alertas são classificados em falsos positivos utilizando diferentes combinações de algoritmos como SOM em conjunto com *K-means* e SVM (*Support Vector Machine*) em conjunto com árvores de decisão. Em Zomlot *et al.* [49] técnicas de aprendizado de máquina também são utilizadas para classificar de forma automatizada grafos de correlação gerados pela ferramenta SnIPS. A abordagem utiliza a técnica SVM para auxiliar o administrador da rede a determinar se os grafos de

correlação são suspeitos ou gerados por falsos positivos.

Em trabalhos mais recente de correlação, Spathoulas e Katsikas [50] realizam a correlação de alertas por meio da similaridade entre os atributos dos alertas e apresenta os grupos formados em um plano cartesiano. Cada grupo exibe características visuais representando métricas derivadas. Por exemplo, a espessura das linhas que formam o grupo no plano representam sua densidade. Já as cores do grupo, representam o quão perigosos são seus possíveis danos à rede.

Seguindo a linha dos trabalhos de Julisch, Zong *et al.* [51] estendem o conceito de causa raiz e propõem um algoritmo eficiente para o problema NP-completo de mineração de alertas críticos: encontrar um número k de causas raiz dos alertas, de modo que a quantidade de alertas disparados em consequência delas seja maximizada. Em outras palavras, o objetivo da abordagem é descobrir as k causas raiz que são responsáveis pela maior quantidade de alertas, já que a solução das causas raiz irá reduzir um volume maior de alertas. No entanto, a abordagem não se restringe a apenas alertas de IDS e considera todo e qualquer tipo de alerta gerado por sistemas de monitoramento e análise.

Chen *et al.* [52], assim como os trabalhos de Njogu e Jiawei [43], utilizam a clusterização para reduzir o volume de alertas de IDS. A abordagem proposta utiliza um algoritmo de seleção de atributos para remover atributos considerados ruídos dos alertas e aprimorar os resultados da clusterização, por meio do algoritmo EM.

Em trabalhos recentes que utilizam técnicas de mineração de dados na correção de alertas, Xuewei *et al.* [9] adotam uma abordagem que combina técnicas de clusterização e cadeias de Markov para representação das dependências causais entre os alertas. Em GhasemiGol e Ghaemi-Bafghi [53], a correlação dos alertas é realizada utilizando o conceito estatístico de entropia. A abordagem defende que as informações contidas em grandes volumes de alertas podem ser representadas por um número menor de hiper-alertas. O método proposto realiza cálculos de entropia sobre os atributos dos alertas e grupos são formados utilizando a técnica de clusterização *DBSCAN* (*Density-Based Spatial Clustering of Applications with Noise*).

Para sumarizar os trabalhos apresentados, a Tabela 2.1 traz um comparativo entre os trabalhos discutidos anteriormente. A tabela apresenta os trabalhos em ordem cronológica que são classificados utilizando os seguintes critérios: i) qual é a abordagem utilizada pelo trabalho, ii) quais são os principais objetivos do trabalho e iii) qual conjunto de dados utilizado para a validação da abordagem. Com relação aos dados utilizados na validação, os trabalhos em que os alertas são provenientes do monitoramento de uma rede real específica são classificados como “Reais”. Os trabalhos em que os alertas são gerados a partir de ataques realizados manualmente são classificados como “Simulação”. Por fim, os trabalhos que fornecem poucas ou nenhuma informação sobre os alertas utilizados na validação da abordagem são classificados como “Não especificado”.

Tabela 2.1 – Tabela comparativa com o resumo dos trabalhos levantados.

Autores	**Ano**	**Abordagem utilizada**	**Objetivo**	**Validação**
Julisch e Dacier [32]	2002	Preprocessamento de alertas	Identificação e remoção das causas raiz dos alertas	Reais
Ning *et al.* [34]	2002	Correlação de alertas com ênfase em modelos visuais	Extração de estratégias de ataque (baseada em pré-requisitos e consequências de intrusão)	2000 DARPA (LLDOS 1.0 e LLDOS 2.0.2)
Julisch [33]	2003	Preprocessamento de alertas	Identificação e remoção das causas raiz dos alertas	Reais
Ning e Xu [6]	2003	Correlação de alertas com ênfase em modelos visuais	Extração de estratégias de ataque (baseada em pré-requisitos e consequências de intrusão)	2000 DARPA (LLDOS 1.0 e LLDOS 2.0.2); Simulação
Treinen e Thurimella [35]	2006	Correlação de alertas com ênfase em modelos visuais	Extração de estratégias de ataque (baseada em regras de associação)	Reais
Lee *et al.* [36]	2006	Correlação de alertas	Correlação probabilística para redução do volume de alertas e extração de informações sobre os ataques	Simulação
Zurutuza *et al.* [37]	2007	Correlação de alertas	Extração de estratégias de ataque (baseada em regras de associação)	2000 DARPA; Simulação
Soleimani e Ghorbani [38]	2008	Correlação de alertas	Extração de alertas críticos e ataques multiestágio (baseada em técnicas de aprendizado de máquina)	2000 DARPA (LLDOS 1.0)
Sadoddin e Ghorbani [39]	2009	Correlação de alertas com ênfase em modelos visuais	Redução do volume de alertas e extração de estratégias de ataque (baseada em técnicas de mineração de padrões frequentes)	2000 DARPA (LLDOS 1.0 e LLDOS 2.0.2); Simulação

Dasireddy *et al.* [40]	2010	Correlação de alertas com ênfase em modelos visuais	Ferramenta para visualização de alertas de baixo nível e identificação de ataques	Reais
Yang *et al.* [41]	2010	Correlação de alertas com ênfase em modelos visuais	Ferramenta para visualização de alertas de baixo nível e identificação de ataques	Não especificado
Liu *et al.* [42]	2010	Correlação de alertas com ênfase em modelos visuais	Extração de cenários de ataque críticos (baseada em autômatos finitos não-determinísticos)	Simulação
Njogu e Jiawei [43]	2010	Correlação de alertas	Correlação para redução do volume de alertas	1999 DARPA; Simulação
Taha *et al.* [44]	2010	Correlação de alertas	Correlação para redução do volume de alertas	1999 DARPA; 2000 DARPA; CTV; DEFCON; Rome AFRL; Honeypot; Treasure Hunt
Cipriano *et al.* [45]	2011	Correlação de alertas	Correlação para previsão de ataques futuros	2008 UCSB International Capture The Flag (iCTF)
Saad e Traore [46]	2011	Correlação de alertas	Correlação para redução do volume de alertas	Simulação; 2000 DARPA; Treasure Hunt
Ahmadinejad *et al.* [47]	2011	Correlação de alertas com ênfase em modelos visuais	Extração de estratégias de ataque (baseada em grafos de ataque e correlação probabilística)	2000 DARPA (LLDOS 1.0 e LLDOS 2.0.2); Simulação; Inter-Service Academy Cyber Defense Exercise (CDX)
Lagzian *et al.* [8]	2012	Correlação de alertas com ênfase em modelos visuais	Extração de estratégias de ataque multiestágio (baseada em técnicas de mineração de padrões frequentes)	2000 DARPA (LLDOS 1.0 e LLDOS 2.0.2)
Fatma e Mohamed [48]	2013	Preprocessamento/Correlação de alertas	Correlação para redução do volume de alertas e redução de alertas falsos positivos	1999 DARPA

Zomlot *et al.* [49]	2013	Correlação de alertas	Classificação de grafos de correlação gerados pela ferramenta SnIPS para identificação de alertas relevantes	Real
Spathoulas e Katsikas [50]	2013	Correlação de alertas com ênfase em modelos visuais	Correlação para representação visual dos alertas e seus possíveis danos à rede	2000 DARPA (LLDOS 1.0); Simulação
Zong *et al.* [51]	2014	Preprocessamento de alertas	Identificação de alertas críticos (causa raiz)	–
Chen *et al.* [52]	2014	Correlação de alertas	Correlação para redução do volume de alertas	2000 DARPA (LLDOS 2.0.2); SANS' incidents.org certification database
Xuewei *et al.* [9]	2014	Correlação de alertas com ênfase em modelos visuais	Correlação para representação visual das causalidades dos alertas utilizando cadeias de Markov	2000 DARPA (LLDOS 1.0)
GhasemiGol e Ghaemi-Bafghi [53]	2014	Correlação de alertas com ênfase em modelos visuais	Correlação para redução do volume de alertas	2000 DARPA (LLDOS 1.0 e LLDOS 2.0.2)

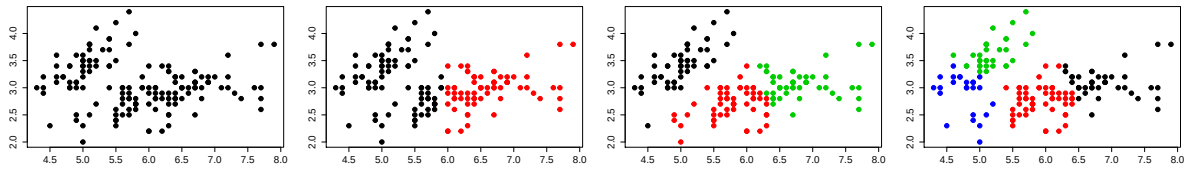
Nota-se que a grande parte dos trabalhos apresentados aborda o problema do grande volume de alertas gerados pelos IDSs, utilizando técnicas de correlação de alertas. A correlação de alertas é utilizada nos trabalhos com diferentes objetivos, por exemplo, para reduzir a quantidade de alertas para análise (hiper-alertas), extrair estratégias de ataque ou identificar alertas críticos. De maneira geral, os trabalhos de correlação buscam identificar similaridades entre os alertas considerando características comuns entre eles. Dessa maneira, os alertas podem ser organizados em grupos que compartilham características semelhantes e a análise pode ser realizada nos grupos formados. As formas de similaridade mais comum encontradas nos trabalhos são: i) similaridade com base nos atributos dos alertas, ii) similaridade com base na relação entre os pré-requisitos e consequências da intrusão e iii) similaridade com base em cenários de ataque conhecidos, muitas vezes chamados de padrões. As técnicas mais comuns utilizadas para realizar o agrupamento dos alertas são as técnicas de clusterização.

A abordagem proposta neste trabalho realiza a correlação de alertas com ênfase em modelos visuais. Com esse objetivo, a abordagem realiza a agregação dos alertas com base em seus atributos, para organizar os alertas em grupos que podem estar relacionados a um mesmo cenário ou estratégia de ataque. Um dos benefícios da abordagem proposta em relação aos trabalhos levantados é que, entre os que utilizam modelos visuais de alto nível para representação das estratégias de ataque, nenhum considera a complexidade dos modelos gerados. Modelos muito grandes podem dificultar e comprometer sua análise e extração de informações, o que vai contra o propósito das abordagens. Outro ponto a ser notado, está relacionado ao conjunto de dados utilizados na validação dos trabalhos. Diferente da abordagem proposta, muito dos trabalhos utilizam um conjunto de alertas antigo, como os conjunto de alertas do DARPA (*Defense Advanced Research Projects Agency*), que remetem a cenários de ataques de muitos anos atrás. O aumento no volume do tráfego da rede, o desenvolvimento de novas aplicações e ataques e as mudanças no comportamento dos usuários [54], faz com que o comportamento dos atacantes atualmente seja diferente do comportamento na época. Além disso, o conjunto de dados do DARPA não pode ser considerado um conjunto de alertas reais, uma vez que foi gerado a partir de um ambiente simulado. Conjuntos de dados gerados por meio de simulação também são adotados na validação de alguns trabalhos. Um ponto negativo nesse aspecto é que, na simulação, as etapas de um ataque são conhecidas, o que pode ajudar na obtenção de bons resultados.

2.3 Clusterização de dados

A clusterização denota o estudo formal de métodos e algoritmos que têm como objetivo organizar um conjunto de dados em grupos denominados de *clusters*. A clusterização busca encontrar uma divisão para os dados, de modo que dados pertencentes a

um mesmo *cluster* sejam similares entre si e dados pertences a *clusters* diferentes sejam dissimilares [11, 55]. A Figura 2.4 traz um exemplo de clusterização.



(a) Conjunto de dados de entrada. (b) Dados organizados em dois *clusters*. (c) Dados organizados em três *clusters*. (d) Dados organizados em quatro *clusters*.

Figura 2.4 – Exemplo de clusterização de um conjunto de dados organizados em diferentes números de *clusters*. Cada *cluster* é representado por uma cor diferente.

Na Figura 2.4, um conjunto de dados inicial é organizado em diferentes números de *clusters*. Os resultados dessas organizações são apresentadas pelas Figuras 2.4b, 2.4c e 2.4d. Cada *cluster* formado é representado por uma cor diferente: a Figura 2.4b apresenta o resultado da clusterização dos dados em dois *clusters*, a Figura 2.4c em três *clusters* e a Figura 2.4d em quatro *clusters*.

A organização dos dados com base em sua similaridade pode ajudar na compreensão e extração de informações relevantes, sobretudo em grandes volumes de dados. Dessa forma, a clusterização é muito utilizada na análise exploratória de dados e empregada em diferentes áreas de conhecimento como estatística, aprendizado de máquina e mineração de dados [56]. Dentre essas áreas, a clusterização tem inúmeras aplicações, por exemplo, no reconhecimento de padrões, segmentação de imagens, recuperação de informação e mineração de textos [11, 55].

Na clusterização alguns aspectos são importantes e fundamentais para o processo de organização e definição dos *clusters*. Um deles é a definição da medida de similaridade entre os pares de dados do conjunto que será clusterizado. A medida de similaridade descreve de forma quantitativa o quão similares os pares de dados do conjunto são. Quanto maior for a medida de similaridade entre os pares de dados, maior a semelhança entre eles. Uma das formas para se quantificar a similaridade entre os pares de dados é por meio de um cálculo da distância ou dissimilaridade [11].

Formalmente, uma medida de distância em um conjunto de dados X é chamada de métrica. Uma métrica é uma função de distância $D(x_i, x_j)$ definida no conjunto X, se para todo $x_i, x_j, x_k \in X$ as seguintes condições forem satisfeitas [56, 57]:

1. $D(x_i, x_j) \geq 0$ (Positividade)
2. $D(x_i, x_j) = D(x_j, x_i)$ (Simetria)
3. $D(x_i, x_j) \leq D(x_i, x_k) + D(x_k, x_j)$ (Desigualdade triangular)

4. $D(x_i, x_j) = 0$, se e somente se, $x_i = x_j$ (Reflexiva)

Os algoritmos de clusterização baseiam-se nas medidas de similaridade ou dissimilaridade entre os dados. Se não houver nenhuma definição de medida de similaridade entre eles, não é possível realizar a clusterização de forma adequada [58]. Além disso, as medidas de similaridades devem ser definidas de forma apropriada para o domínio do conjunto de dados. Por exemplo, uma das formas mais simples para quantificar a distância entre dados com atributos numéricos é por meio da distância Euclidiana, caso especial ($p = 2$) da métrica de Minkowski [11]. Sejam $x = (x_1, x_2, \ldots, x_d)$ e $y = (y_1, y_2, \ldots, y_d)$ dois dados formados por d atributos ou características representados no espaço euclidiano d-dimensional, a distância Euclidiana é definida como:

$$D(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

A distância Euclidiana é comumente utilizada para avaliar a proximidade entre pares de dados numéricos representados em duas ou três dimensões [11]. No entanto, para dados em que os valores de seus atributos não são numéricos, como dados com atributos qualitativos, a distância Euclidiana torna-se inapropriada como medida de dissimilaridade e outras medidas capazes de quantificar de maneira mais adequada a proximidade entre os pares de dados devem ser adotadas. Um exemplo são dados de presença/ausência, também conhecidos como dados binários. Um dado de presença/ausência $X = \{0,1\}^p$ é descrito por um conjunto p de características, em que assume-se o valor 1 no caso em que o dado possui determinada característica e o valor 0 caso contrário [12]. A Tabela 2.2 apresenta um exemplo de um conjunto de dados binários representados por vinte e duas características. Os dados são conhecidos como vetores de características.

Tabela 2.2 – Dados de presença/ausência (binários) representados por 22 características.

	Características																					
Dados	**c1**	**c2**	**c3**	**c4**	**c5**	**c6**	**c7**	**c8**	**c9**	**c10**	**c11**	**c12**	**c13**	**c14**	**c15**	**c16**	**c17**	**c18**	**c19**	**c20**	**c21**	**c22**
A	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	1	0	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1

Existem diferentes medidas de distâncias propostas para quantificar a similaridade entre dados binários. As medidas, de modo geral, quantificam a proximidade entre os dados levando em conta o número de características comuns entre eles e o número de características distintas. O índice de similaridade de Jaccard ou coeficiente de Jaccard é um exemplo de medida de similaridade para dados binários que considera essas caracte-

rísticas. Sejam A e B dois dados binários com n atributos, o índice de similaridade de Jaccard é dado por:

$$J(A, B) = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} \tag{2.1}$$

tal que:

- M_{11} representa o número total de características comuns a ambos os dados.
- M_{01} representa o número total de características ausentes em A mas não em B.
- M_{10} representa o número total de características presentes em A mas não em B.

A distância de Jaccard, que mede a dissimilaridade entre dois conjunto é dada por:

$$D_J(A, B) = 1 - J(A, B) \quad = \frac{M_{01} + M_{10}}{M_{01} + M_{10} + M_{11}} \tag{2.2}$$

Por exemplo, entre as amostras A e B da Tabela 2.2, duas características (c1 e c3) ocorrem em ambos os dados, isto é, $M_{11} = 2$. Duas características ocorrem em B mas não ocorrem em A (c5 e c6), assim $M_{01} = 2$. Por fim, duas características ocorrem em A mas não ocorrem em B (c2 e c4), logo $M_{10} = 2$. Portanto, de acordo com o índice de similaridade de Jaccard (Equação 2.1), a similaridade entre os dados binários A e B é de $J(A, B) = 2/(2 + 2 + 2) = 2/6 = 0{,}3333333$. A medida de dissimilaridade entre os dados (Equação 2.2) é $D_J = 1 - J(A, B) = 1 - 0{,}3333333 = 0{,}6666667$.

2.3.1 Análise de clusterização hierárquica

Existem diferentes métodos de clusterização que, de modo geral, podem ser classificados em duas categorias: métodos por particionamento e métodos hierárquicos [11]. Nos algoritmos de clusterização que utilizam métodos por particionamento, um conjunto inicial de dados é dividido em k grupos, em que o número de grupos k pode ser previamente determinado, por exemplo, o algoritmo de clusterização *K-means* [11]. Já nos algoritmos que utilizam métodos hierárquicos, os grupos são formados por meio de sucessivos agrupamentos/divisões, criando uma hierarquia de *clusters* que é representada por uma árvore de *clusters* ou dendrograma. O dendrograma é uma representação estruturada em árvore nas quais os nós folhas representam os *clusters* formados por um único dado e os ramos da árvore representam os dados que são agrupados em cada iteração do algoritmo formando novos *clusters*. Os ramos do dendrograma ascendem até o nó raiz, nos quais todos os dados do conjunto irão compor o mesmo *cluster*. Um exemplo de um dendrograma é apresentado pela Figura 2.5.

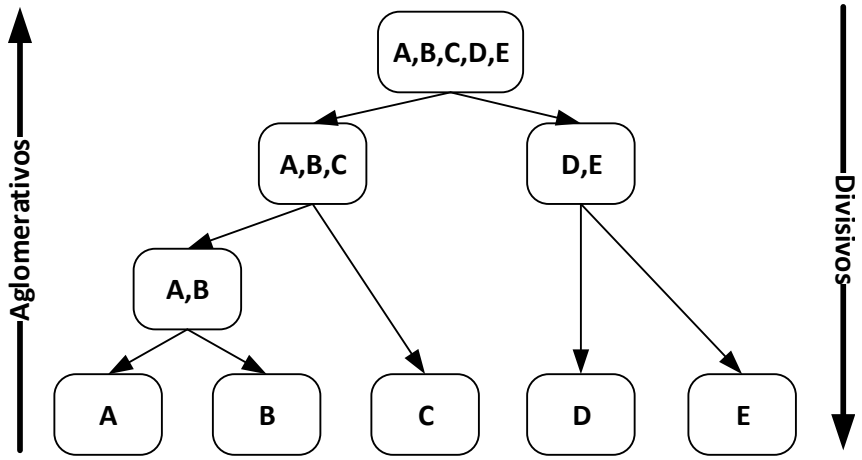


Figura 2.5 – Exemplo de uma árvore de *clusters* ou dendrograma. Adaptado de [2].

Os algoritmos de clusterização hierárquica podem adotar diferentes abordagens no processo de formação dos *clusters*. Os algoritmos divisivos utilizam uma abordagem *top-down*, enquanto que os algoritmos aglomerativos utilizam uma abordagem *bottom-up*. Nas abordagens divisivas, inicialmente todos os n dados do conjunto são agrupados em um único *cluster*. Em seguida, a cada iteração do algoritmo o *cluster* é dividido, formando novos *clusters* com menos dados. O processo se repete até que sejam formados n *clusters* contendo um dado cada. Por outro lado, nas abordagens aglomerativas, inicialmente são formados n *clusters* contendo apenas um dado. Em cada iteração do algoritmo, os *clusters* são unidos até formar um único *cluster* com n dados [59]. Os métodos divisivos e aglomerativos podem ser visualizados no dendrograma da Figura 2.5.

2.3.2 Exemplo de clusterização utilizando o método aglomerativo

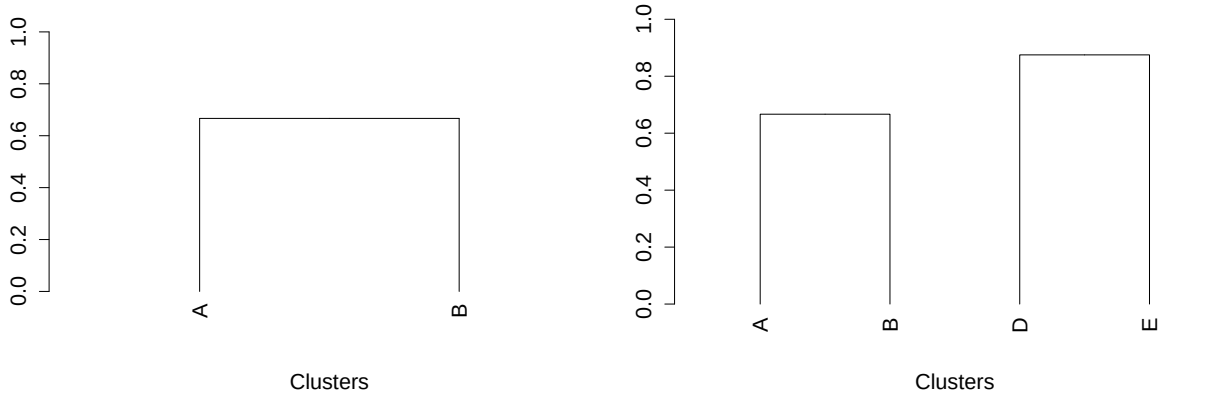
O ponto de partida para os algoritmos de clusterização hierárquica é a chamada matriz de dissimilaridade. A matriz de dissimilaridade mede a distância entre os pares de dados de determinado conjunto. A Tabela 2.3 apresenta uma matriz de dissimilaridade obtida considerando os pares de amostras apresentadas na Tabela 2.2, em que medida de similaridade entre os dados utilizada foi a distância de Jaccard (Equação 2.2). A tabela é uma matriz simétrica $A = (a_{ij})$ de ordem m, tal que m representa o número total de dados e $i, j \in \{1, 2, \ldots, m\}$ representam os dados $1, 2, \ldots, m$, respectivamente. A diagonal principal da matriz representa as medidas de dissimilaridade de um dado para si mesmo, logo, todas as medidas da diagonal principal possuem valor zero, isto é, $a_{ij} = 0$ se $i = j \ \forall i, j \in \{1, 2, \ldots, m\}$. Na Tabela 2.3, ambas as medidas de dissimilaridade da região triangular superior e inferior da matriz são apresentadas. As medidas de dissimilaridade da região triangular inferior são apresentadas em destaque.

A partir da matriz de dissimilaridade, o primeiro passo realizado pelo algoritmo de clusterização hierárquica é procurar os pares de dados que são mais similares, ou seja, os

Tabela 2.3 – Matriz de dissimilaridade entre os pares de amostras da Tabela 2.2 com base no distância de Jaccard.

Amostras	**A**	**B**	**C**	**D**	**E**
A	0,0000000	0,6666667	0,8461538	1,0000000	1,0000000
B	0,6666667	0,0000000	0,9285714	1,0000000	1,0000000
C	0,8461538	0,9285714	0,0000000	1,0000000	1,0000000
D	1,0000000	1,0000000	1,0000000	0,0000000	0,8750000
E	1,0000000	1,0000000	1,0000000	0,8750000	0,0000000

que possuem a menor medida de dissimilaridade. Na Tabela 2.3, os pares de dados mais similares são os pares A e B com medida de dissimilaridade igual a 0,6666667. Ambos os pares são então agrupadas em um *cluster*, que será representado por um nó no nível 0,6666667 de um dendrograma, como mostra a Figura 2.6a.



(a) Primeira iteração da clusterização hierárquica. (b) Segunda iteração da clusterização hierárquica.

Figura 2.6 – Dendrogramas resultantes das duas primeiras iterações da clusterização hierárquica utilizando o método do vizinho mais distante (*Complete Linkage).*

A próxima etapa do algoritmo é similar à etapa anterior. No entanto, agora é preciso definir qual o método utilizado para realizar o cálculo de similaridade entre o *cluster* formado (A,B) em relação aos outros dados do conjunto. O método definido é chamado de método de ligação (*linkage method*) e determina qual o tipo de clusterização hierárquica será executada pelo algoritmo. Existem diversos métodos de ligação que podem ser utilizados pelo algoritmo de clusterização hierárquica. As técnicas mais amplamente utilizadas são as que utilizam a abordagem aglomerativa [2]. Entre as abordagens aglomerativas, a maioria dos algoritmos de clusterização hierárquica são variantes dos métodos *Single Linkage*, *Complete Linkage* e Método de *Ward* [11]. Por exemplo, no método *Single Linkage* ou vizinho mais próximo, a distância entre dois *clusters* é definida como a menor distância entre os pares de dados, um pertencendo a cada *cluster*. Já o método *Complete Linkage* ou vizinho mais distante considera a maior distância entre eles. A Tabela 2.4 apresenta um comparativo entre os métodos aglomerativos mais utilizados e como é calculada a dissimilaridade entre os pares de *clusters* em cada um deles.

Tabela 2.4 – Métodos de ligação para o cálculo de dissimilaridade entre os *clusters* na clusterização hierárquica.

Método	**Descrição**	**Fórmula**
Single Linkage	Distância mínima ou vizinho mais próximo. A distância entre os pares de *clusters* é dada pela menor distância entre os pares de dados pertencentes a *clusters* diferentes.	$min\ \{D(a,b) : a \in A, b \in B\}$
Complete Linkage	Distância máxima ou vizinho mais distante. A distância entre os pares de *clusters* é dada pela maior distância entre os pares de dados pertencentes a *clusters* diferentes.	$max\ \{D(a,b) : a \in A, b \in B\}$
Average Linkage	A distância entre os pares de *clusters* é dada pela média da distância entre todos os pares de dados pertencentes a cada um dos *clusters*.	$\frac{1}{\|A\|\|B\|}\sum_{a\in A}\sum_{b\in B} D(a,b)$
Método de *Ward*	Mínima variância ou soma dos erros quadráticos. A distância entre os pares de *clusters* é dada pela soma dos quadrados entre todos os dados pertencentes a cada um dos *clusters*. Os pares de *clusters* com o menor aumento na soma global dos quadrados dentro dos *clusters* serão agrupados.	$\frac{n_A n_B}{n_A + n_B}\|\vec{m}_A - \vec{m}_B\|^2$ onde $\vec{m}_j$ é o centro do *cluster* j, e n_j é o números de dados contidos no *cluster*.

A Tabela 2.5 apresenta a dissimilaridade calculada entre o *cluster* (A,B) e os outros pares de dados utilizando o método de clusterização *Complete Linkage*. Por exemplo, a dissimilaridade entre as amostras A e C é igual a 0,8461538, enquanto que a dissimilaridade entre as amostras B e C é igual a 0,9285714. Portanto, o método *Complete Linkage* seleciona o valor máximo de dissimilaridade entre os pares de amostras, ou seja, 0,9285714, para quantificar a dissimilaridade entre (A,B) e C. Os outros valores de dissimilaridade da matriz são obtidos de forma análoga.

Tabela 2.5 – Matriz de dissimilaridade após o agrupamento entre os *clusters* A e B, utilizando o método *Complete Linkage* para atualizar os valores de dissimilaridade do novo *cluster* em relação aos demais.

Amostras	**(A,B)**	**C**	**D**	**E**
(A,B)	0,0000000	0,9285714	1,0000000	1,0000000
C	0,9285714	0,0000000	1,0000000	1,0000000
D	1,0000000	1,0000000	0,0000000	0,8750000
E	1,0000000	1,0000000	0,8750000	0,0000000

O processo agora se repete. Utilizando a matriz de dissimilaridade apresentada na Tabela 2.5, o algoritmo de clusterização hierárquica irá agrupar os *clusters* D e E por serem os pares de *clusters* mais similares. Eles serão representados por um nó no nível 0,8750000 do dendrograma, como mostra a Figura 2.6b. Em seguida, os valores de dissimilaridade

entre o *cluster* formado (D,E) e os demais são recalculados. Por exemplo, a dissimilaridade entre o *cluster* (A,B) e (D,E) é o valor máximo entre $D_J(A,(D,E)) = 1{,}0000000$ e $D_J(B,(D,E)) = 1{,}0000000$. A matriz de dissimilaridade resultante é apresentada pela Tabela 2.6.

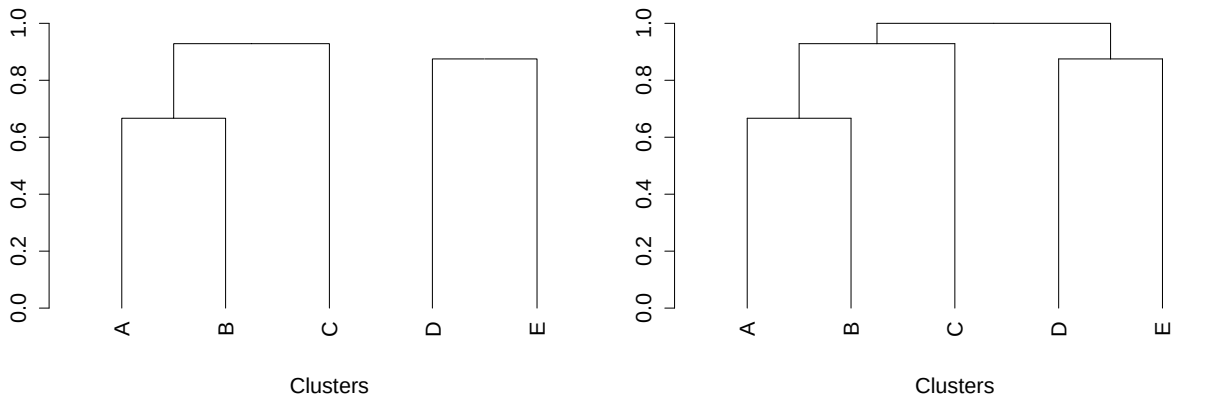
Tabela 2.6 – Matriz de dissimilaridade após o agrupamento entre os *clusters* D e E. O *cluster* resultante da iteração é o *cluster* (D,E).

Amostras	**(A,B)**	**C**	**(D,E)**
(A,B)	0,0000000	0,9285714	1,0000000
C	0,9285714	0,0000000	1,0000000
(D,E)	1,0000000	1,0000000	0,0000000

Os pares de *clusters* mais similares na Tabela 2.6 são os pares (A,B) e C com similaridade igual a 0,9285714. Os pares são então agrupados e representados por um novo nó no nível 0,9285714 do dendrograma como mostra a Figura 2.7a. Na próxima etapa, os valores de similaridade entre os novo *cluster* formado (A,B,C) e os demais são recalculados, resultando na matriz de dissimilaridade apresentada na Tabela 2.7. Finalmente, os dois *clusters* restantes (A,B,C) e (D,E) são agrupados e representados pelo nó raiz do dendrograma no nível 1,0000000. A execução do algoritmo é finalizada e o seu retorno é o dendrograma apresentado pela Figura 2.7b.

Tabela 2.7 – Matriz de dissimilaridade após o agrupamento entre os *clusters* (A,B) e C. O *cluster* resultante da iteração é o *cluster* (A,B,C).

Amostras	**(A,B,C)**	**(D,E)**
(A,B,C)	0,0000000	1,0000000
(D,E)	1,0000000	0,0000000



(a) Terceira iteração da clusterização hierárquica. (b) Quarta iteração da clusterização hierárquica.

Figura 2.7 – Dendrogramas resultantes das duas últimas iterações da clusterização hierárquica utilizando o método do vizinho mais distante (*Complete Linkage*).

A análise dos resultados da clusterização hierárquica por meio do dendrograma da Figura 2.7b é bastante direta. Nele é possível observar que os dados mais similares são

os pares de dados (A,B), seguidos pelos pares (D,E). De fato, os pares de dados (A,B) possuem duas características em comum, enquanto que os pares (D,E) apresentam apenas uma. Apesar dos pares (A,C) também compartilharem duas características em comum e (B,C) compartilharem uma, o dado C possui muitas características presentes que são ausentes em A e B, o que faz com que a dissimilaridade entre os pares seja maior. A partir da análise, é possível determinar qual o número de *clusters* que deseja-se obter. Cortes na horizontal do dendrograma representam diferentes organizações dos grupos dos dados. Quanto mais próximo do nós folhas for o corte, mais similares são os dados pertencentes a um *cluster*. Por outro lado, quanto mais próximo o corte for da raiz, mais dissimilares são os dados. Com base nessa análise é possível determinar qual o número de *clusters* que deseja-se obter de acordo com o problema.

2.4 Mineração de processos

A mineração de processos descreve um conjunto de métodos e abordagens que combinam técnicas de mineração de dados com modelagem de processos de negócios (BPM - *Business Process Modeling*). O objetivo da mineração de processos é sintetizar modelos de processos, extraindo conhecimento de informações registradas em um log por um sistema de informação. A abordagem busca descobrir modelos de processos que descrevem com precisão como ocorrem os processos de fato (em possível discordância entre como foram projetados para ocorrer), examinando informações sobre sua execução em um log. Nesse contexto, os modelos de processos podem ser utilizados como ferramentas de apoio na análise e monitoramento de processos, bem como para o aperfeiçoamento e reprojeto de processos existentes.

O ponto de partida das técnicas de mineração de processos são os logs. Para a mineração de processos, cada registro encontrado no log é considerado um evento, razão pela qual os logs são conhecidos como log de eventos. Além disso, é importante que os logs de eventos contenham informações adequadas sobre as atividades executadas no processo, uma vez que elas serão consideradas por algoritmos de mineração de processos para construção dos modelos. Portanto, para extrair informações dos logs de eventos, algumas características devem ser consideradas [3]:

- Cada evento no log corresponde a uma *atividade*. A atividade de um evento representa uma ação (ou etapa) realizada no processo que foi registrada no log. Por exemplo, em um sistema de cadastro de usuários, *criar usuário*, *atualizar usuário* e *remover usuário* são exemplos de atividades de eventos registrados em um log.

- Eventos podem ter atributos como *atividade*, *tempo* e *recurso*. O atributo *atividade* descreve a ação relacionada ao evento, como mencionado anteriormente. O atributo

tempo registra o *timestamp* (data e/ou hora) do evento. Por fim, o atributo *recurso* descreve o responsável pela execução da atividade.

- Um processo pode ser executado mais de uma vez. Além disso, os eventos que compõem duas execuções do mesmo processo podem ser diferentes. Cada execução do processo (isto é, cada instância do processo) representa um *case*. Portanto, cada evento no processo deve pertencer a um *case*, ou seja, a uma instância de execução do processo. Em um sistema de cadastro de usuários, os eventos relacionados a um determinado usuário podem compor um *case*, por exemplo.

- Eventos de um mesmo *case* devem ser ordenados de maneira crescente, por exemplo, por ordem de ocorrência. A ordem dos eventos é muito importante, já que ela pode alterar as dependências causais entre os eventos, alterando os resultados de alguns algoritmos.

A Tabela 2.8 traz um exemplo de um log de eventos que apresenta essas características.

Tabela 2.8 – Exemplo de um log de eventos. Extraído e adaptado de [3].

Id do *Case*	**Id do Evento**	**Propriedades**				
		Timestamp	**Atividade**	**Recurso**	**Custo**	**...**
1	35654423	30-12-2010:11.02	Pedido de registro	Pete	50	...
	35654424	31-12-2010:10.06	Examinar cuidadosamente	Sue	400	...
	35654425	05-01-2011:15.12	Verificar o bilhete	Mike	100	...
	35654426	06-01-2011:11.18	Decidir	Sara	200	...
	35654427	07-01-2011:14.24	Rejeitar o pedido	Pete	200	...
2	35654483	30-12-2010:11.32	Pedido de registro	Mike	50	...
	35654485	30-12-2010:12.12	Verificar o bilhete	Mike	100	...
	35654487	30-12-2010:14.16	Examinar casualmente	Pete	400	...
	35654488	05-01-2011:11.22	Decidir	Sara	200	...
	35654489	08-01-2011:12.05	Pagar indenização	Ellen	200	...
3	35654521	30-12-2010:14.32	Pedido de registro	Pete	50	...
	35654522	30-12-2010:15.06	Examinar casualmente	Mike	400	...
	35654524	30-12-2010:16.34	Verificar o bilhete	Ellen	100	...
	35654525	06-01-2011:09.18	Decidir	Sara	200	...
	35654526	06-01-2011:12.18	Reiniciar o pedido	Sara	200	...
	35654527	06-01-2011:13.06	Examinar cuidadosamente	Sean	400	...
	35654530	08-01-2011:11.43	Verificar o bilhete	Pete	100	...
	35654531	09-01-2011:09.55	Decidir	Sara	200	...
	35654533	15-01-2011:10.45	Pagar indenização	Ellen	200	...
...	...	...	...	...	...	...

Cada entrada registrada no log representa um evento relacionado ao tratamento de pedidos de indenização. Cada evento está associado a uma instância de execução do processo ou *case*. Neste exemplo, os eventos estão associados a três *cases* (*Case* 1, 2 ou 3). Cada evento possui um identificador e quatro propriedades: *Timestamp*, Atividade, Recurso e Custo. No contexto da mineração de processos, as propriedades são denominadas atributos. Cada evento está relacionado a uma ação como *Pedido de registro, Examinar cuidadosamente* e *Decidir*, que é representada pelo atributo atividade. Além disso, dentro de um mesmo *case*, os eventos estão ordenados de forma crescente por meio do atributo *timestamp*, que registra o tempo da ocorrência de cada evento.

Existem três grandes áreas relacionadas à mineração de processos: descoberta de processos, verificação de conformidade e aperfeiçoamento do modelo [3]. A descoberta de processos está relacionada à transformação de um log de eventos em um modelo de processo. Técnicas de descoberta de processos recebem como entrada um log de eventos e retornam como saída um modelo de processo, de maneira que o modelo seja uma representação precisa do comportamento observado no log. A verificação de conformidade faz uso de métricas como *fitness* e *precision* para avaliar a qualidade do modelo de processo gerado no contexto do log. Por fim, o aperfeiçoamento do modelo utiliza novas informações para enriquecer o modelo descoberto, explorando novas perspectivas em relação às informações extraídas dos logs como as perspectivas organizacional e temporal.

No próximo capítulo, é apresentada a abordagem proposta neste trabalho que utiliza as técnicas de mineração de processos e da clusterização hierárquica para sintetizar o grande volume de alertas de IDS em modelos visuais de alto nível.

3 ANÁLISE DE ALERTAS DE INTRUSÃO BASEADA NA MINERAÇÃO DE PROCESSOS

Neste capítulo, é apresentada a abordagem proposta neste trabalho que tem como objetivo minerar alertas de intrusão para obter conhecimento sobre o comportamento dos atacantes. A abordagem proposta utiliza a mineração de processos e técnicas de clusterização hierárquica para extrair informações sobre as estratégias de ataque multiestágio utilizadas por eles a fim de comprometer a rede. Então, as estratégias de ataque são representadas em modelos visuais e apresentadas para o administrador da rede.

O ponto de partida da abordagem proposta são os alertas de IDS que registram informações sobre as tentativas de ataque contra uma rede ou sistema computacional. A abordagem visa extrair informações contidas no grande volume de alertas IDS para analisá-los e descobrir novas informações e relações entre os ataques que antes não podiam ser inferidas. No entanto, a utilização de alertas de IDS possui algumas limitações. Devido a natureza de seus dados, os alertas podem apresentar ruídos e imperfeições que podem trazer alguns desafios para sua análise [60, 61]. Para entender essas limitações, é preciso distinguir os conceitos de eventos, alertas e incidente de segurança.

No contexto da segurança dos computadores, eventos são entidades de baixo nível, não necessariamente maliciosos, que são analisadas pelos IDSs como um pacote de rede ou uma chamada de sistema. Um alerta é uma entidade gerada pelo IDS para notificar o administrador da rede sobre eventos que são anormais ou suspeitos. Assim, um único evento pode causar vários alertas, enquanto que um único alerta pode representar um conjunto ou sequência de eventos [62]. Um incidente de segurança é um evento malicioso confirmado que viola os princípios da segurança da informação [61]. IDSs geram alertas para quaisquer eventos que ocorrem em uma rede e podem incluir alertas que não são provenientes de atividades maliciosas ou ataques. Da mesma maneira, alguns ataques podem passar pelo IDS e nenhum alerta ser gerado. Consequentemente, uma vez que a abordagem proposta neste trabalho tem como base os alertas de IDS, ela é suscetível a essas limitações.

Outro aspecto importante acerca dos alertas considerados na abordagem está relacionado às informações de classificação do ataque detectado. Somente por meio da classificação do ataque é possível descrever quais foram as ações executadas pelo atacante e representar suas estratégias de ataque. Por exemplo, em alertas de IDS baseado em assinaturas, a detecção dos ataques é realizada por meio da identificação de características ou padrões dos ataques no tráfego que está sendo analisado. Dessa forma, por meio dos padrões (chamados de assinatura) o ataque detectado pode ser classificado (ataque do tipo *Buffer Overflow*, por exemplo) e suas informações podem ser registradas no alerta.

Para a utilização de alertas gerados por outras categorias de IDS, como no caso de alertas de IDSs baseados em anomalias, é necessário que após a detecção da intrusão, o ataque seja classificado e as informações sejam registradas no alerta. A classificação dos ataques é uma informação necessária para a abordagem proposta neste trabalho.

3.1 Metodologia

A abordagem proposta é dividida em quatro etapas que consistem em transformar as informações dos alertas de baixo nível em modelos visuais de alto nível conforme mostra a Figura 3.1.

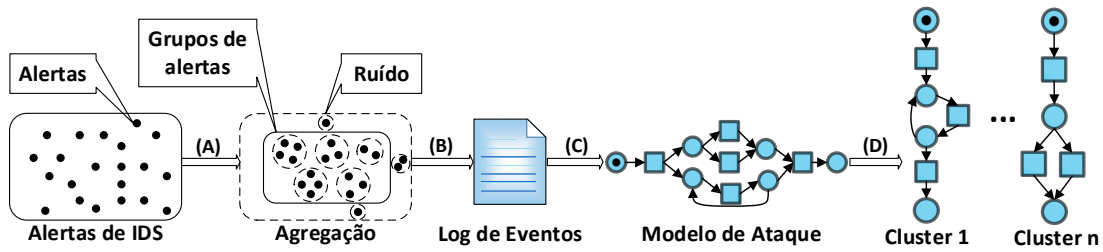


Figura 3.1 – As quatro etapas da abordagem proposta.

Na primeira etapa da abordagem (Figura 3.1 (A)), alertas que compartilham características em comum são agregados. Em seguida, na segunda etapa (Figura 3.1 (B)), os alertas agregados na etapa anterior são representados em um formato adequado para a mineração de processos. A seguir, na terceira etapa (Figura 3.1 (C)), um algoritmo é utilizado para realizar a descoberta dos modelos de ataque que representam os comportamentos dos atacantes encontrados nos alertas. Por fim, na última etapa (Figura 3.1 (D)), é realizada a análise nos modelos resultantes, em que técnicas de clusterização hierárquica são aplicadas em modelos complexos de difícil compreensão com o objetivo de torná-los mais simples e intuitivos. Nas próximas seções, cada etapa da abordagem é descrita em detalhes.

3.1.1 Agregação dos alertas de IDS

Alertas de IDSs coletam informações que podem variar de acordo com o tipo de dispositivo (baseado em *host* ou rede), método de detecção (baseado em assinaturas ou anomalias) e até mesmo de acordo com o fabricante do dispositivo. Alguns exemplos de informações que podem ser registradas nos alertas incluem o endereço IP de origem/destino, portas de origem/destino, o *timestamp* de quando ocorreu o ataque, informações do número de sistema autônomo (ASN - *Autonomous System Number*), a severidade do ataque, qual o protocolo violado, entre outros. A Tabela 3.1 mostra um exemplo de alertas de IDS

baseado em assinatura que apresentam algumas dessas informações, também chamadas de atributos dos alertas.

Tabela 3.1 – Exemplo de algumas informações registradas nos alertas de IDS.

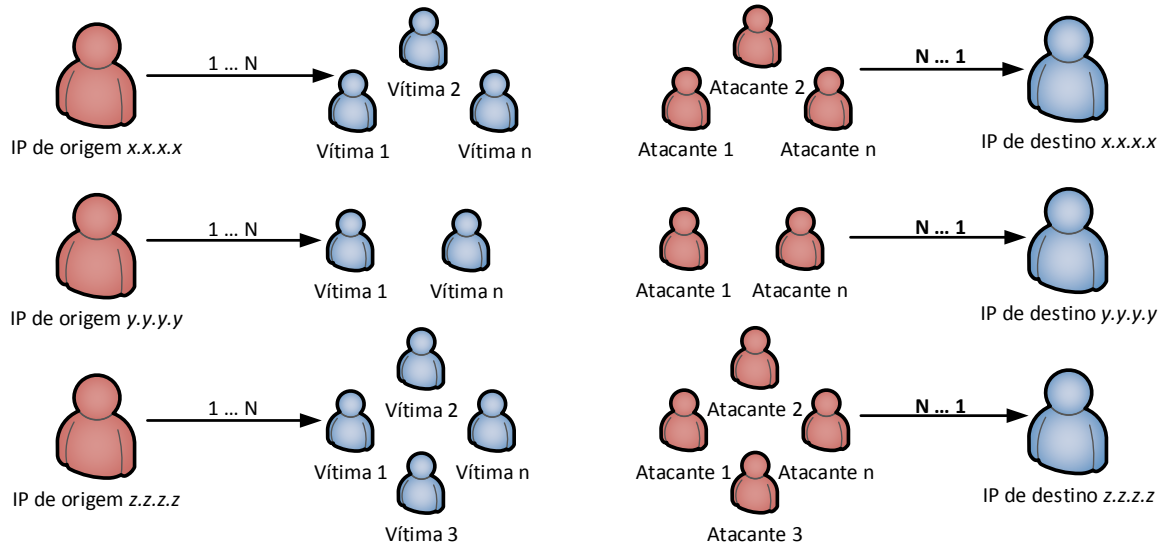
Timestamp	**Protocolo**	**Assinatura**	**Severidade**	**IP de origem**	**...**
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 20:46:54	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 20:58:34	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 20:47:44	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 20:53:48	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 20:56:54	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 21:05:04	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 20:55:48	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 21:02:00	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 20:43:08	HTTP	Shell Command Execution (root.exe)	4	*y.y.y.y*	...
2012-08-01 15:06:49	Invalid TCP Traffic	Possible nmap Scan (XMAS (FIN PSH URG))	3	*z.z.z.z*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 15:06:49	Invalid TCP Traffic	Impossible Flags (SFRPAU)	3	*z.z.z.z*	...
2012-08-01 20:53:48	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 15:06:49	Invalid TCP Traffic	Impossible Flags (SFRPAU)	3	*z.z.z.z*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 20:49:08	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 20:45:08	HTTP	Shell Command Execution (root.exe)	4	*y.y.y.y*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 20:59:50	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 15:05:25	Invalid TCP Traffic	Possible nmap Scan (XMAS (FIN PSH URG))	3	*z.z.z.z*	...
2012-08-01 21:06:00	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...
2012-08-01 15:07:25	Nmap scanner	FUP OS Fingerprinting Probe	3	*z.z.z.z*	...
2012-08-01 01:00:29	MS-RPC	DCOM ISystemActivator Overflow	1	*x.x.x.x*	...
2012-08-01 21:03:00	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 20:51:08	HTTP	Nimda Attack (root.exe)	1	*y.y.y.y*	...
2012-08-01 01:00:29	SMB	ASN.1 Bitstring Processing Heap Overflow	1	*x.x.x.x*	...

Na análise de alertas de intrusão, o objetivo é obter conhecimento sobre o comportamento dos atacantes e descobrir quais estratégias eles estão utilizando para tentar comprometer a rede. Após a descoberta das estratégias, os administradores da rede terão conhecimento sobre a sequência de passos, além das dependências entre os passos tomados

pelos atacantes nas execuções dos ataques. Em vista disso, na primeira etapa da abordagem proposta, alertas com características semelhantes são agregados visando agrupar os alertas que podem estar relacionados a um mesmo ataque ou a um mesmo conjunto de ataques. Em seguida, nas próximas etapas da abordagem, o relacionamento entre esses alertas será melhor investigado no processo de descoberta dos modelos.

A agregação dos alertas pode ser realizada de diferentes maneiras. Cada agregação visa agrupar os alertas de intrusão sob diferentes perspectivas em relação aos ataques. Algumas formas mais comuns de agregação são [36, 45]:

- **Agregação um-para-muitos**: os alertas são organizados de acordo com o endereço IP de origem do alerta formando grupos em que um único endereço IP de origem tenta comprometer muitos endereços IP de destino, como mostra a Figura 3.2a.
- **Agregação muitos-para-um**: os alertas são organizados de acordo com o endereço IP de destino do alerta formando grupos em que muitos endereços IP de origem tentam comprometer um único endereço IP de destino, como mostra a Figura 3.2b.



(a) Agregação um-para-muitos.

(b) Agregação muitos-para-um.

Figura 3.2 – Diferentes formas para agregação dos alertas de IDS.

Por exemplo, para investigar as estratégias de ataque empregadas pelos atacantes, pode-se agregar os alertas que compartilham o mesmo endereço IP de origem (agregação um-para-muitos). Desta maneira, cada grupo de alertas formado está associado às ações de um mesmo atacante, que, posteriormente será investigado para descobrir quais são suas estratégias utilizadas. De maneira similar, para investigar as estratégias de ataques distribuídos contra a rede, pode-se agregar os alertas que compartilham o mesmo endereço

IP de destino (agregação muitos-para-um). Nesse caso, a agregação visa agrupar alertas que representam como diferentes atacantes colaborativamente tentaram comprometer uma mesma vítima. A *island-hopping* [45], que visa formar grupos de alertas provenientes de ataques em que o atacante utiliza uma vítima previamente comprometida para realizar outros ataques, como em uma *botnet*, é outro exemplo de agregação. A flexibilidade de representar diferentes perspectivas na agregação dos alertas de IDS pode ser explorada pelo administrador da rede para fornecer uma visão mais abrangente sobre a rede em relação aos ataques ocorridos.

Após formar os grupos de alertas seguindo determinado critério de agregação, um processo de filtragem é realizado. O objetivo da filtragem é identificar grupos de alertas que não estão relacionados a ataques multiestágio e não correspondem ao comportamento geral da rede. Portanto, os grupos são considerados ruídos pela abordagem. Nesse sentido, dois grupos de alertas são descartados: i) grupos formados por um único alerta e ii) grupos formados somente por alertas associados à mesma assinatura. Os grupos formados por um único alerta representam ataques que se encerram logo após a execução da violação. Por outro lado, os grupos formados por alertas associados à mesma assinatura são semelhantes aos grupos formados por um único alerta. No entanto, após o início do ataque, o ataque entra em laço antes de encerrar sua execução. A Figura 3.3 apresenta o comportamento dos ataques gerado pelos grupos de alertas descartados pela abordagem.

(a) Comportamento dos ataques dos grupos formados por um único alerta.

(b) Comportamento dos ataques dos grupos de alertas associados à mesma assinatura.

Figura 3.3 – Estratégias de ataque dos grupos de alertas descartados.

Na Figura 3.3, em ambos os casos, a sequência de passos do ataque realizado pelos atacantes envolve apenas uma etapa. Esse cenário difere de ataques multiestágio, que são o foco da abordagem proposta, nos quais os atacantes executam uma sequência de etapas (estratégia de ataque) a fim de atingir seu objetivo. Portanto, os grupos de alertas que representam essas situações tem baixa prioridade ao avaliarmos os comportamentos mais frequentes e gerais da rede. Logo, esses grupos de alertas são descartados e esses comportamentos não são incluídos no modelo gerado.

3.1.2 Conversão dos alertas agregados em um log de eventos

Conforme definido na Seção 2.4, para mineração de processos, o conjunto de dados de entrada consiste de eventos registrados em um log. Uma vez que a abordagem proposta neste trabalho faz o uso da mineração de processos para a análise de alertas de IDS, a segunda etapa da abordagem consiste em converter os alertas agregados na etapa anterior em um log de eventos. Conforme apresentado na Tabela 3.1, os alertas de IDS podem possuir diversos atributos. No entanto, nem todos os atributos são necessários para que seja possível descobrir as estratégias de ataque utilizados pelos atacantes. Dessa forma, uma seleção de atributos pode ser realizado nos alertas, mantendo apenas os atributos que fornecem informações sobre a ocorrência dos ataques. Então, essas informações serão utilizadas para construir um log de eventos, atendendo as características necessárias para a mineração de processos como os conceitos de *case* e atividade dos eventos (Seção 2.4).

Primeiramente, é definida a atividade do evento, ou seja, as ações dos atacantes que são reportadas por meio das assinaturas dos alertas de IDS. A atividade do evento é uma informação importante, pois será representada pelos vértices no modelo gerado. Os vértices do modelo representam a sequência de passos realizados nos ataques e ajudam na identificação e visualização das dependências entre as etapas que o constituem. Conforme mencionado no início do Capítulo 3, a informação sobre a classificação do ataque detectado pelo IDS é necessária na abordagem proposta. Essa informação é utilizada para definir a atividade dos eventos.

Em um log de eventos, os eventos devem pertencer a um *case*, isto é, a uma instância de execução do processo. Durante a descoberta do modelo de processo, os *cases* são utilizados para determinar quais as dependências causais entre as atividades executadas no log de eventos. No caso de ações executadas em um log de eventos de alertas de IDS, os *cases* são utilizados para determinar as dependências causais entre as etapas de um ataque. Na abordagem proposta, é definida como um *case* cada grupo de alertas formado na primeira etapa da abordagem (Seção 3.1.1) que ocorreram dentro de um intervalo de tempo t. Por exemplo, suponha que sejam agregados alertas de acordo com o endereço IP de origem (um-para-muitos) e com intervalo de tempo $t = 1$ dia. Assim, todos alertas com endereço IP de origem *x.x.x.x* disparados no dia m irão pertencer ao *case i*. Todos alertas com endereço IP *x.x.x.x* disparados no dia n irão pertencer ao *case j*. Por fim, todos alertas com endereço IP de origem *y.y.y.y* disparados no dia m irão pertencer ao *case k*. Dessa maneira, cada atacante compõe uma instância do processo e o conjunto de alertas disparados por suas ações, que ocorreram dentro do intervalo $t = 1$ dia, são os eventos que compõem o *case*. Finalmente, em um log de eventos, os eventos pertencentes a um *case* devem ser ordenados por ocorrência. No contexto dos alertas de IDS, é utilizada a informação do *timestamp* para ordenar os alertas.

Como exemplo, a Tabela 3.2 mostra o resultado do processo de conversão dos

alertas da Tabela 3.1 para um log de eventos.

Tabela 3.2 – Log de eventos de alertas de IDS apresentados na Tabela 3.1.

Instância do processo (Id do *Case*)	**Propriedades**		
	Timestamp	**Atividade**	**Recurso**
1	2012-08-01 20:43:08	Shell Command Execution (root.exe)	*y.y.y.y*
	2012-08-01 20:45:08	Shell Command Execution (root.exe)	*y.y.y.y*
	2012-08-01 20:46:54	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 20:47:44	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 20:49:08	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 20:51:08	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 20:53:48	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 20:53:48	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 20:55:48	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 20:56:54	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 20:58:34	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 20:59:50	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 21:02:00	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 21:03:00	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 21:05:04	Nimda Attack (root.exe)	*y.y.y.y*
	2012-08-01 21:06:00	Nimda Attack (root.exe)	*y.y.y.y*
2	2012-08-01 01:00:29	DCOM ISystemActivator Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
	2012-08-01 01:00:29	ASN.1 Bitstring Processing Heap Overflow	*x.x.x.x*
3	2012-08-01 15:05:25	Possible nmap Scan (XMAS (FIN PSH URG))	*z.z.z.z*
	2012-08-01 15:06:49	Possible nmap Scan (XMAS (FIN PSH URG))	*z.z.z.z*
	2012-08-01 15:06:49	Impossible Flags (SFRPAU)	*z.z.z.z*
	2012-08-01 15:06:49	Impossible Flags (SFRPAU)	*z.z.z.z*
	2012-08-01 15:07:25	FUP OS Fingerprinting Probe	*z.z.z.z*

Os alertas foram agregados de acordo com o endereço IP de origem e com intervalo de tempo $t = 1$ dia. Foram filtrados os atributos dos alertas, mantendo apenas a assinatura do alerta, atribuída como atividade do evento, o endereço IP de origem, atribuído como o

recurso do evento, e o *timestamp* para ordenar os alertas por ordem de ocorrência. Foram formados três *cases*. Cada *case* é composto por eventos ordenados associados as ações de um mesmo atacante. A partir do log de eventos de alertas de IDS, é possível realizar o processo de descoberta do modelo de ataque. Esse processo será apresentado na próxima seção.

3.1.3 Descoberta do modelo de ataque

O processo de construção do modelo de ataque é uma etapa importante na abordagem proposta. Nesta etapa, é preciso definir o algoritmo de descoberta que receberá como entrada um log de eventos de alertas de IDS definido na etapa anterior, e irá gerar como saída um modelo que representa o comportamento dos atacantes encontrado no log. Na abordagem proposta, os modelos são gerados a partir do Algoritmo 3.1.

Algoritmo 3.1 Pseudo código para o algoritmo de descoberta do modelo

Entrada:
L : Log de eventos de alertas de IDS
Saída:
Um modelo de ataque G

```
 1: function ModelDiscovery(L)
 2:     Adicionar vértice V_i em G
 3:     Adicionar vértice V_f em G
 4:     for all case c ∈ L do
 5:         Seja A[1...k] um array com todas as k atividades ∈ c
 6:         // Em todos os casos, se o vértice ou a aresta existir, sua frequência é incrementada
 7:         Adicionar vértice A[1] em G
 8:         Adicionar aresta (V_i, A[1]) em G
 9:         for i ← 2 até k do
10:             Adicionar vértice A[i] em G
11:             Adicionar aresta (A[i − 1], A[i]) em G
12:             if i = k then
13:                 Adicionar aresta (A[i], V_f)
14:             end if
15:         end for
16:     end for
17:     return (G)
18: end function
```

O Algoritmo 3.1 recebe como entrada um log de eventos de alertas de IDS e tem como saída um modelo definido pela 4-upla (V, E, V_i, V_f). O modelo é dado por um conjunto de vértices V, um conjunto de arestas direcionadas E, um vértice inicial V_i e um vértice final V_f. Com exceção dos vértices V_i e V_f, todos os vértices V e arestas E do modelo possuem um peso associado que representa a frequência absoluta da ocorrência de determinada atividade (no caso dos vértices) ou da transição entre duas atividades (no caso das arestas) no modelo. O Algoritmo 3.1 opera da seguinte maneira. Inicialmente, são inseridos no modelo os vértices V_i e V_f (linhas 2 e 3). Em seguida (laço das linhas 4 à 16),

para cada *case* do log, vértices são inseridos no modelo para cada uma de suas atividades. Uma aresta é inserida conectando os vértice V_i e o vértice que representa a atividade inicial do *case* (linhas 7 e 8). Os vértices que representam atividades subsequentes no *case* são conectados por arestas (laço das linhas 9 à 15). Caso um vértice ou aresta entre dois vértices já exista no modelo, sua frequência é incrementada em 1. Por fim, é inserida uma aresta conectando o vértice que representa a atividade final do *case* e V_f (linhas 12 e 13). Ao término da execução do algoritmo, o modelo é retornado.

Como exemplo, aplicando o Algoritmo 3.1 no log de alertas de IDS da Tabela 3.2 obtém-se como resultado o modelo apresentado na Figura 3.4. O log é bastante simples com apenas 3 *cases* e 7 atividades distintas. O modelo apresenta 9 vértices, um para cada atividade distinta do log mais os vértices inicial e final, e 15 arestas. Analisando o modelo, é possível observar as sequências de passos que foram executados pelos atacantes.

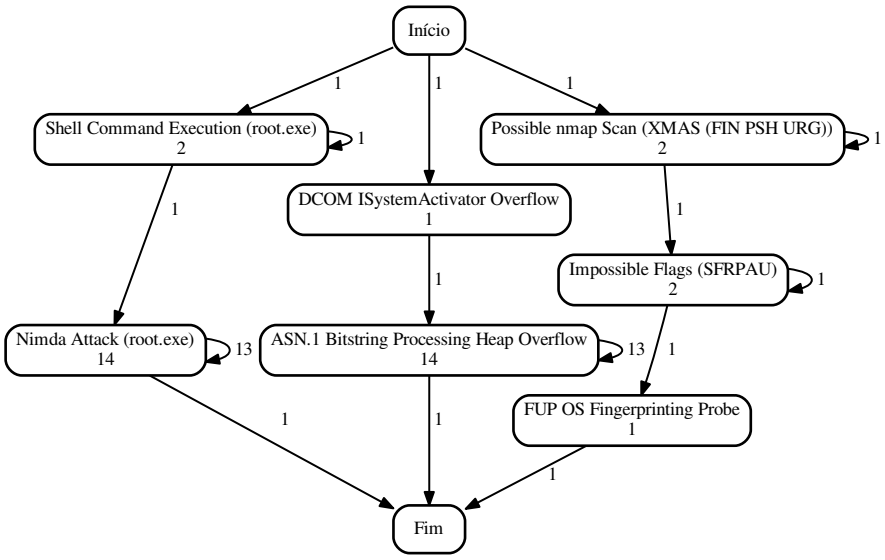


Figura 3.4 – Modelo de ataque gerado utilizando o Algoritmo 3.1 no log de alertas de IDS da Tabela 3.2.

Por exemplo, na sequência de ataque representada pelos vértices à esquerda no modelo, o atacante primeiramente acessa via comando *Shell* o arquivo *root.exe*. O arquivo indica que o sistema foi previamente infectado pelos *worms Code Red II* ou *sadmind/IIS* que criam um *backdoor* no sistema, permitindo a execução de ataques remotos, como reportado pelo CERT/CC (*Computer Emergency Response Team Coordination Center*) Advisory CA-2001-06 [63]. Em seguida, com o sistema vulnerável, o atacante executa o *worm Nimda*. *Worms* são uma classe de *malware* (programas maliciosos) autônomos que utilizam a rede como meio de propagação para infectar outros *hosts*, realizando uma cópia de si mesmo no sistema vulnerável. Na sequência de ataque representada pelos vértices no centro do modelo, o atacante tenta explorar uma vulnerabilidade do tipo *Buffer Overflow* na tecnologia de comunicação entre processos RPC (*Remote Procedure Call*) da Microsoft. Em seguida, uma vulnerabilidade do tipo *Integer Overflow* no ASN.1 (*Abstract*

Syntax Notation One) da Microsoft é explorada. Os dois ataques, caso explorados com sucesso, permitem a execução de códigos remotos nas vítimas e podem fornecer acesso não autorizado ao atacante, como reportado pelos CVE (*Common Vulnerabilities and Exposures*) 2003-0352 e 2003-0818 [64, 65]. Já na sequência de passos representada pelos vértices à direita no modelo, o atacante executa três ataques de reconhecimento, que têm como objetivo coletar informações sobre as vítimas e sobre suas vulnerabilidades.

Além da análise que pode ser realizada para entender sobre os ataques que estão sendo empregados contra a rede, por meio do modelo, é possível observar as frequências das atividades e as frequências das transições entre atividades. Por exemplo, o *Nimda Attack (root.exe)* e *ASN.1 Bitstring Processing Heap Overflow* foram os ataques mais executados (14 vezes). Analisando a Tabela 3.2, é fácil verificar que ambas atividades, de fato, foram as que mais ocorreram. Essas informações podem auxiliar o administrador da rede a priorizar as ações protetivas contra os ataques.

O modelo descoberto na Figura 3.4 é um modelo bastante simples e muitas de suas informações podem ser verificadas diretamente no log de alertas. No entanto, nem todos modelos descobertos são assim. No contexto da análise de alertas de intrusão, devido a grande quantidade de alertas disparados pelos IDSs, os modelos gerados podem se tornar grandes e complexos, impossibilitando que sua análise seja realizada facilmente. Por esse motivo, a clusterização automática de modelos complexos utilizando técnicas de clusterização hierárquica é realizada na última etapa da abordagem que é apresentada na próxima seção.

3.1.4 Clusterização automática de modelos complexos

Uma das preocupações da abordagem proposta está relacionada à complexidade dos modelos gerados. Muitas vezes, os modelos de ataque podem se tornar grandes e complexos devido à natureza não estruturada dos *cases* que pertencem ao log de alertas. Por exemplo, a não homogeneidade entre as atividades dos *cases* é um dos fatores que contribuem para o aumento de complexidade dos modelos. Um exemplo de não homogeneidade ocorre quando, em um log de eventos, há um grande número de *cases*, em que quase todos os *cases* possuem um comportamento distinto de qualquer outro. Em outras palavras, os *cases* são disjuntos em relação as atividades executadas pelos atacantes, isto é, eles compartilham poucas ou nenhuma atividades em comum, como os *cases* 1, 2 e 3 na Tabela 3.2. Uma vez que, de acordo o Algoritmo 3.1, os vértices no modelo representam as atividades encontradas no log de alertas, a não homogeneidade entre as atividades dos *cases* faz com que o número de atividades distintas aumente e, consequentemente, o número de vértices no modelo também aumenta. Além disso, cria-se no modelo novos caminhos (novas arestas) para os *cases* o que faz com que o modelo aumente em tamanho e complexidade. A Figura 3.5 apresenta um exemplo de um modelo complexo.

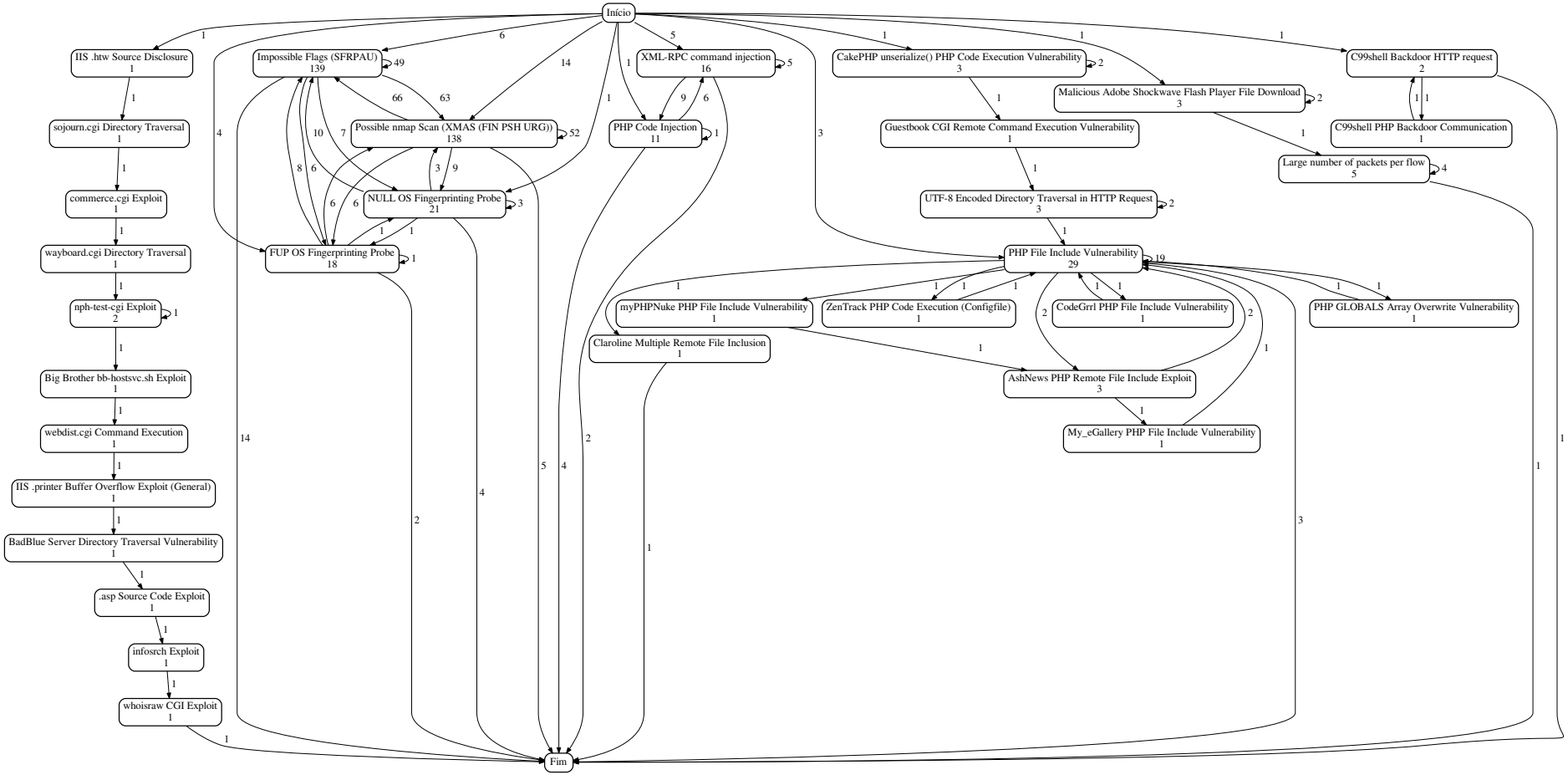


Figura 3.5 – Exemplo de um modelo complexo.

O modelo da Figura 3.5 apresenta 35 vértices e 78 arestas. Devido ao grande número de vértices e arestas, muitos fluxos de ataque são criados no modelo o que dificulta realizar a análise visual. Dessa modo, o objetivo da abordagem proposta pode ser prejudicado, já que o administrador de rede teria dificuldades em visualizar as estratégias de ataque utilizadas pelos atacantes.

Uma das possíveis soluções para o problema dos modelos complexos é reordenar os *cases*, organizando-os em grupos de *cases* que possuem atividades em comum (e são mais homogêneos), e grupos de *cases* que possuem poucas ou nenhuma atividade em comum (e são menos homogêneos). Assim, a complexidade do modelo pode ser reduzida, dividindo o modelo complexo em modelos menores, mais simples e compreensíveis. Para realizar a redução de complexidade do modelos, a abordagem proposta utiliza o Algoritmo 3.2.

Algoritmo 3.2 Pseudo código para o algoritmo Case Clustering

Entrada:
L : Log de eventos com n *cases* e k atividades distintas
Saída:
Um conjunto de modelos $G = \{\ g \mid g \subseteq M(L) \wedge g$ não é complexo $\}$

```
1: function CaseClustering(L)
2:     M(L) ← ModelDiscovery(L)                          ▷ Algoritmo 3.1
3:     if M(L) não é complexo then
4:         return (M(L))
5:     else
6:         D_cases ← DiscretizarCases(L)                 ▷ Algoritmo 3.3
7:         M_dist ← MatrizDissimilaridade(D_cases)
8:         H_clust ← ClusterizaçãoHierárquica(M_dist)
9:         r ← H_clust.raiz()
10:        Seja Q uma fila inicialmente vazia
11:        Q.enqueue(r)
12:        while Q não está vazia do
13:            u ← Q.dequeue()
14:            if u é um cluster folha then
15:                G' ← SingleCaseClustering(u)          ▷ Algoritmo 3.4
16:                G ← G ∪ {G'}
17:            else
18:                for all cluster c adjacentes a u do
19:                    M(c) ← ModelDiscovery(c)
20:                    if M(c) é complexo then
21:                        Q.enqueue(c)
22:                    else
23:                        G ← G ∪ {M(c)}
24:                    end if
25:                end for
26:            end if
27:        end while
28:        return (G)
29:    end if
30: end function
```

A ideia do algoritmo é utilizar a clusterização hierárquica nos *cases* do log de alertas para gerar um dendrograma que vai auxiliar a identificar quais os *cases* do log são mais homogêneos em relação as suas atividades do que outros. Com base no dendrograma, é possível reorganizar os *cases* em grupos ou *clusters* de modo que *cases* homogêneos pertençam ao mesmo *cluster*, e *cases* não homogêneos pertençam a *clusters* diferentes. Então, modelos não complexos são gerados a partir dos *clusters* formados.

O primeiro passo realizado pelo Algoritmo 3.2 é analisar a complexidade do modelo gerado a partir do log de eventos de alertas recebido como entrada. Uma das dificuldades que surge com a utilização da clusterização hierárquica é conseguir determinar o nível de corte no dendrograma, isto é, qual o número de *clusters* necessários para que os modelos gerados por eles não sejam modelos complexos. Por exemplo, para reduzir a complexidade do modelo da Figura 3.5, o modelo precisa ser dividido em quantos modelos? Além disso, como determinar se os modelos provenientes da divisão não são complexos? Outro problema que surge está relacionado em como definir um modelo complexo, ou seja, como determinar se um modelo é compreensível para análise ou se o modelo é complexo e precisa ser clusterizado, sem realizar a análise visual do modelo?

Para atender a essas questões, a abordagem proposta faz uso de uma métrica e um valor de limiar. Por meio da métrica, é possível quantificar a simplicidade de um modelo, e com o valor obtido, avaliar se o modelo é complexo comparando-o com o valor de limiar. Caso a simplicidade do modelo avaliado exceda o valor de limiar, o modelo é considerado complexo, caso contrário não. Isso elimina a subjetividade da análise visual dos modelos para determinar se ele é visualmente complexo, permitindo que essa tarefa seja realizada de forma automatizada. A métrica e o valor de limiar são utilizados pelo Algoritmo 3.2 para realizar a avaliação dos modelos e em conjunto com o dendrograma gerado pela clusterização hierárquica, o algoritmo é capaz de determinar em quantos *clusters* ou divisões são necessárias para que o modelo complexo se torne mais simples. Um outro benefício da utilização da métrica de simplicidade e o valor de limiar é que ambos são independentes do algoritmo, o que possibilita que outras formas de quantificação possam ser utilizadas no futuro. Na abordagem, para um modelo G com V vértices e E arestas, a simplicidade é dada por:

$$\text{Simplicidade}(G) = \frac{|V|}{|E|} \tag{3.1}$$

A definição da simplicidade por meio da relação entre o número de vértices e o número de arestas pode ser explicada por meio da observação dos modelos. Um modelo com 3 vértices, sendo 2 deles os vértices inicial e final, pode ter no máximo 3 arestas: uma aresta conectando o vértice com o vértice inicial V_i, uma aresta conectando o vértice com o vértice final V_f e uma aresta conectando o vértice consigo mesmo. De acordo com a Equação 3.1, o modelo tem simplicidade $3/3 = 1$. Um modelo com 4 vértices pode ter no

máximo 8 arestas e tem simplicidade $4/8 = 0{,}5$. Um modelo com 5 vértices pode ter no máximo 15 arestas e tem simplicidade $5/15 = 0{,}333$. Um modelo com n vértices tem no máximo $((n-2)^2+2\times(n-2)) = (n^2-2n)$ arestas e tem simplicidade igual a $n/(n^2-2n)$. Nota-se que conforme o número de vértices aumenta, maior a possibilidade de conexões entre eles e maior o potencial para novas arestas.

Com base na métrica de simplicidade, o valor de limiar pode ser determinado. A determinação do valor de limiar de simplicidade dos modelos não é uma tarefa trivial e deve ser calculado considerando características dos próprios modelos. Na abordagem, o valor de limiar é calculado da seguinte maneira. Primeiramente, modelos com diferentes números de vértices e arestas são gerados (segundo uma forma de agregação) e sua simplicidade é calculada de acordo com a Equação 3.1. Em seguida, os modelos são classificados como complexos ou não complexos com o auxílio da análise visual de um especialista, como mostra a Tabela 3.3.

Tabela 3.3 – Classificação dos modelos complexos para determinação do valor de limiar.

# de vértices	**# de arestas**	**Simplicidade**	**É visualmente complexo?**
...	...	...	...
10	51	0,196078431	Não
11	37	0,297297297	Não
12	47	0,255319149	Não
13	41	0,317073171	Não
14	40	0,350000000	Não
15	36	0,416666667	Não
17	55	0,309090909	Sim
18	70	0,257142857	Não
21	36	0,583333333	Não
24	46	0,521739130	Não
31	168	0,184523810	Sim
35	120	0,291666667	Sim
36	122	0,295081967	Sim
...	...	...	...

A ideia principal da classificação dos modelos é utilizar o ponto de vista de um especialista para estimar o valor de limiar de simplicidade de outros modelos. Por meio da Tabela 3.3, é possível notar que, nesse caso, os modelos que apresentam um número de vértices menor do que 15 não são complexos. De maneira similar, os modelos que apresentam um número de vértices maior do que 30 frequentemente são classificados como complexos devido ao grande número de arestas que dificultam a visualização e análise do modelo. Com base nessas informações, dado o número de vértices e utilizando como referência a simplicidade (Equação 3.1) dos modelos classificados como não complexos, pode-se estimar o valor de limiar de outros modelos utilizando uma análise de regressão linear. A regressão linear permite investigar a relação entre duas ou mais variáveis e seu

principal objetivo é estimar um valor de uma variável, com base nos valores conhecidos de outras variáveis. Deste modo, a relação entre o número de vértices e a simplicidade (Equação 3.1) dos modelos não complexos (classificados anteriormente e utilizados como referência) é investigada na regressão linear para estimar o valor de limiar de qualquer outro modelo, dado o seu número de vértices. A equação da regressão linear é dada por:

$$Y_k = \alpha + \beta X_k \tag{3.2}$$

onde Y_k é chamada de variável dependente ou o valor que se quer estimar, α é um termo constante que representa a interceptação da reta no eixo vertical, β é um termo constante que representa o coeficiente angular da reta e X_k representa a variável independente. No contexto da abordagem proposta, Y_k representa o valor de limiar de simplicidade que deseja-se estimar para um modelo, dado o seu número de vértices X_k.

Com base na regressão linear e nas informações dos modelos classificados (Tabela 3.3), para um modelo G com V vértices, E arestas, simplicidade S e um valor de limiar Y_k definido pela regressão linear, um exemplo de análise de complexidade do modelo pode ser realizada da seguinte maneira:

$$\text{É complexo}(G) = \begin{cases} \text{Não}, & \text{se } |V| < 15 \text{ ou } S > Y_k \\ \text{Sim}, & \text{se } |V| > 30 \text{ ou } S < Y_k \end{cases} \tag{3.3}$$

Por exemplo, utilizando a configuração apresentado pela Equação 3.3, os modelos que apresentam um número menor do que 15 vértices são classificados como não complexos. Por outro lado, os modelos que apresentam um número maior do que 30 vértices são classificados como complexos. Essas condições podem ser utilizadas por um administrador da rede para configurar preferências em que não se deseja clusterizar modelos com poucos vértices, porém, modelos com muitos vértices devem ser clusterizados independente do valor de limiar. Para modelos que apresentam um número de vértices $15 \leq |V| \leq 30$ o valor de limiar é utilizado. Supondo que o valor de limiar para um modelo com 20 vértices seja $Y_k = 0{,}44$. Um modelo com 20 vértices e 100 arestas, de acordo com a Equação 3.1, apresenta simplicidade igual a $S = 20/100 = 0{,}2 < Y_k$. Logo, uma vez que a simplicidade do modelo é menor do que o valor de limiar permitido (ou seja, o modelo é menos simples ou mais complexo), segundo a Equação 3.3, o modelo avaliado é considerado complexo. Por outro lado, um modelo com 20 vértices e 40 arestas apresenta simplicidade igual a $S = 20/40 = 0{,}5 > Y_k$. Desse modo, o modelo apresenta simplicidade maior do que o valor de limiar (logo, é mais simples ou menos complexo) e é considerado não é complexo. Portanto, com base na análise discutida anteriormente, é possível definir o valor de limiar para o número de vértices (definido a partir da Tabela 3.3) e o número de arestas (definido a partir da regressão linear) de um dado modelo.

Após a definição da análise de complexidade dos modelos, o Algoritmo 3.2 opera do seguinte modo. Primeiramente (linhas 3 e 4), utilizando a métrica de simplicidade e o valor de limiar (Equação 3.3, por exemplo), o algoritmo verifica se o modelo gerado a partir do log de alertas recebido como entrada é complexo. Caso o modelo seja complexo, o primeiro passo realizado pelo algoritmo é aplicar a clusterização hierárquica no modelo para obter o dendrograma com a organização dos *cases*. Para construir o dendrograma, a clusterização hierárquica deve calcular a similaridade entre os *cases* com base em suas atividades utilizando uma medida de distância. Portanto, os *cases* precisam ser representados de tal modo que esse cálculo possa ser realizado. Uma das formas de representação dos *cases* para realização do cálculo de similaridade entre eles é transformá-los em vetores de características binários. Os *cases* são discretizados e transformados em vetores binários, em que cada posição do vetor representa uma atividade distinta do log. Para todas as atividades executadas em um *case*, o valor 1 é atribuído no vetor na posição que representa essa atividade. Às outras posições do vetor, que representam atividades não executadas pelo *case*, é atribuído o valor 0. O resultado é uma matriz como a apresentada na Tabela 2.2, em que os dados são os *cases* e as características são as atividades do log. No Algoritmo 3.2 (linha 6), esse processo é realizado utilizando o Algoritmo 3.3.

Algoritmo 3.3 Pseudo código do para o algoritmo Discretizar Cases

Entrada:
L : Log de eventos com n *cases* e k atividades distintas
Saída:
Uma matriz $M_{n \times k}$ dos *cases* discretizados

```
 1: function DiscretizarCases(L)
 2:     Inicializar cada elemento da matriz M com o valor 0
 3:     Seja C[1...n] um array com todos os cases distintos ∈ L
 4:     Seja A[1...k] um array com todas atividades distintas ∈ L
 5:     for i ← 1 até n do
 6:         for all atividade a ∈ C[i] do
 7:             j ← índice da posição da atividade a em A
 8:             M[i][j] ← 1
 9:         end for
10:     end for
11:     return (M)
12: end function
```

A partir da matriz com os *cases* discretizados, o algoritmo calcula a similaridade entre os *cases* do log (linha 7 do Algoritmo 3.2). A similaridade entre os *cases* é calculada utilizando uma medida de distância para dados binários como a distância de Jaccard (Equação 2.2). O processo tem como retorno uma matriz de dissimilaridade que é utilizada como entrada pelo algoritmo de clusterização hierárquica (linha 8 do Algoritmo 3.2). O algoritmo de clusterização hierárquica, por sua vez, produz como saída um dendrograma. O dendrograma apresenta como os *cases* estão organizando hierarquicamente com base na similaridade entre suas atividades. Conforme mencionado na Seção 2.3.1, em um den-

drograma, os nós folhas representam os *clusters* formados por um único dado. Uma vez que na abordagem os dados que estão sendo clusterizados são os *cases* do log, cada nó folha representa um *cluster* contendo um único *case*. A partir dos nós folhas, os ramos do dendrograma representam os *clusters* que são formados agrupando os *cases* similares até o nó raiz, que representa todos os *cases* organizados em um mesmo *cluster*. Sendo assim, a raiz do dendrograma representa o modelo complexo que deseja-se clusterizar.

Iniciando pela raiz do dendrograma, utilizando uma abordagem *top-down*, a ideia do Algoritmo 3.2 (laço das linhas 12 à 27) é encontrar no dendrograma um conjunto de *clusters* nos quais os modelos gerados pelos *cases* que pertencem a esses *clusters* não sejam complexos. Para isso, uma busca em largura é realizada no dendrograma para avaliar os *clusters* quanto à complexidade dos modelos gerados por seus *cases*. Por exemplo, utilizando a busca em largura a partir da raiz, o algoritmo avalia os *clusters* que formam o próximo nível do dendrograma. Caso os modelos gerados pelos *clusters* sejam complexos, o algoritmo desce um nível no dendrograma e o processo se repete. No entanto, caso seja encontrado um *cluster* que pertence a esse nível e seu modelo não seja complexo, não é necessário mais avaliar esse ramo do dendrograma, apenas os ramos do dendrograma em que os *clusters* geram modelos complexos. Dessa forma, no fim da busca, um conjunto de *clusters* em que o modelo complexo inicial deve ser dividido é encontrado e retornado.

A busca em largura avalia os *clusters* a partir da raiz até os nós folhas que são os últimos *clusters* a serem analisados. No entanto, o que ocorre quando um modelo gerado por um *cluster* folha ainda é complexo? Em outras palavras, como realizar o processo de clusterização em modelos complexos que são gerados a partir de um único *case*? Na abordagem, a solução para esses casos é apresentada pelo Algoritmo 3.4 (invocado na linha 15 do Algoritmo 3.2).

Para reduzir a complexidade dos modelos gerados por um único *case*, a abordagem reorganiza os eventos (alertas) que compõem o *case* em grupos de eventos considerando a duração em que os ataques ocorreram. Dessa maneira, é possível dividir o *case* em grupos de eventos com base em seu *timestamp*. Por exemplo, se o intervalo de tempo entre o primeiro e o último evento do *case* é de 30 minutos, os eventos que pertencem ao *case* podem ser divididos em dois grupos com duração de 15 minutos cada. Os eventos que compõe o primeiro grupo são os que ocorreram nos primeiros 15 minutos do ataque e os que compõem o segundo grupo são os que ocorreram nos últimos 15 minutos do ataque. Se os modelos produzidos pelos grupos ainda são complexos, os eventos podem ser reagrupados e divididos novamente em três grupos com duração de 10 minutos cada. Um ponto importante a ser notado é que, uma vez que os eventos de um *case* são organizados por ordem de ocorrência de acordo com o *timestamp* do alerta (Seção 2.4), a formação dos grupos deve manter essa ordem. Caso a ordem não seja mantida, os modelos gerados pelos grupos de alertas podem combinar violações que ocorreram em etapas distintas do

Algoritmo 3.4 Pseudo código do para o algoritmo Single Case Clustering

Entrada:
L : Log de eventos incluindo um único *case*
Saída:
Um conjunto de modelos $G = \{ g \mid g \subseteq M(L) \wedge g$ não é complexo $\}$

```
1: function SingleCaseClustering(L)
2:     n ← 2
3:     repeat
4:         Seja E_L um conjunto com todos os grupos formados após a divisão dos eventos do
       case em n intervalos de tempo
5:         Seja i um grupo de eventos ∈ E_L
6:         if ∀i ∈ E_L | M(i) não é complexo then
7:             G ← G ∪ {M(i)}
8:             return (G)
9:         else if ∃i ∈ E_L | M(i) não é complexo ∧ {E_L} \ {i} mantém a ordem dos eventos
       then
10:            G ← G ∪ {M(i)}
11:            E_L ← {E_L} \ {i}
12:            n ← 2
13:        else
14:            n ← n + 1
15:        end if
16:    until seja possível subdividir L
17:    return (G)
18: end function
```

ataque. Supondo que os eventos de um *case* que teve duração de 30 minutos (intervalo entre o primeiro e o último evento do *case*) foram divididos em três grupos t_1, t_2 e t_3. Os eventos de cada grupo ocorrem em um intervalo de 10 minutos cada, de modo que o *timestamp* dos eventos do grupo $t_1 < t_2 < t_3$. Se o modelo gerado pelos grupos t_1 e t_3 são complexos, mas o modelo gerado pelo grupo t_2 não o é, no contexto da abordagem, os grupos $\{t_1, t_3\}$ são reagrupados e divididos novamente com base em um intervalo de tempo. No entanto, ao realizar o reagrupamento, os eventos que ocorreram no início da execução do ataque (t_1) e os eventos que ocorreram em seu fim (t_3) são combinados, o que não pode ocorrer. Portanto, $\{t_1, t_2, t_3\}$ são reagrupados e divididos em intervalos menores (quatro intervalos de 7 minutos e 30 segundos, por exemplo) e os novos grupos formados são reavaliados. Por outro lado, se o modelo gerado pelos grupos t_1 e/ou t_3 não forem complexos, mas o modelo de t_2 o for, os eventos do grupo $\{t_1, t_2\}$ ou $\{t_2, t_3\}$ são reagrupados e o mesmo problema não ocorre. Por fim, é importante notar que a divisão dos grupos de eventos, na pior das hipóteses, ocorre até que a duração dos ataques seja maior do que um segundo. Caso contrário, não é mais possível realizar a divisão dos eventos em intervalos e, consequentemente, reduzir a complexidade dos modelos com base no *timestamp* dos alertas.

Considerando as situações mencionadas anteriormente, o Algoritmo 3.4 opera da seguinte maneira. O algoritmo inicializa o número de intervalos em que os grupos de

eventos que pertencem ao *case* serão divididos (linha 2). Em seguida (laço das linhas 3 à 16), os eventos são subdivididos em grupos com base em seu *timestamp*, considerando o número de intervalos de tempo definido. Após a divisão dos eventos nos grupos de intervalos (linha 4), se todos os modelos gerados a partir dos grupos de eventos formados não são complexos (condição da linha 6), o conjunto de modelos é retornado pelo algoritmo. No entanto, se existe um grupo de eventos t em que o modelo gerado a partir dele não é complexo, e a ordem dos eventos do *case* é mantida após reagrupar os eventos dos grupos restantes que geram modelos complexos (condição da linha 9, conforme discutido anteriormente), não é mais necessário dividir t e os grupos de eventos restantes são reagrupados e o processo se repete a partir desses grupos (linhas 10 à 12). Por fim (linhas 13 e 14), se após a divisão dos eventos nos grupos de intervalos, todos os modelos gerados a partir dos grupos de eventos são complexos, o número de intervalos para divisão dos grupos é incrementado e o processo se repete.

Como exemplo, aplicando o Algoritmo 3.2 para reduzir a complexidade do modelo da Figura 3.5 apresentado no início da seção, foi necessário dividir o modelo complexo em outros quatro modelos. A Figura 3.6 apresenta o resultado da cluterização. Os *clusters* formados estão representados por áreas em destaque. Cada uma das áreas representa um modelo que será apresentado para o administrador da rede em substituição do modelo complexo.

No próximo capítulo, é realizada a validação da abordagem proposta e um estudo de caso é realizado para testar a abordagem utilizando um conjunto real de alertas de IDS da Universidade de *Maryland*.

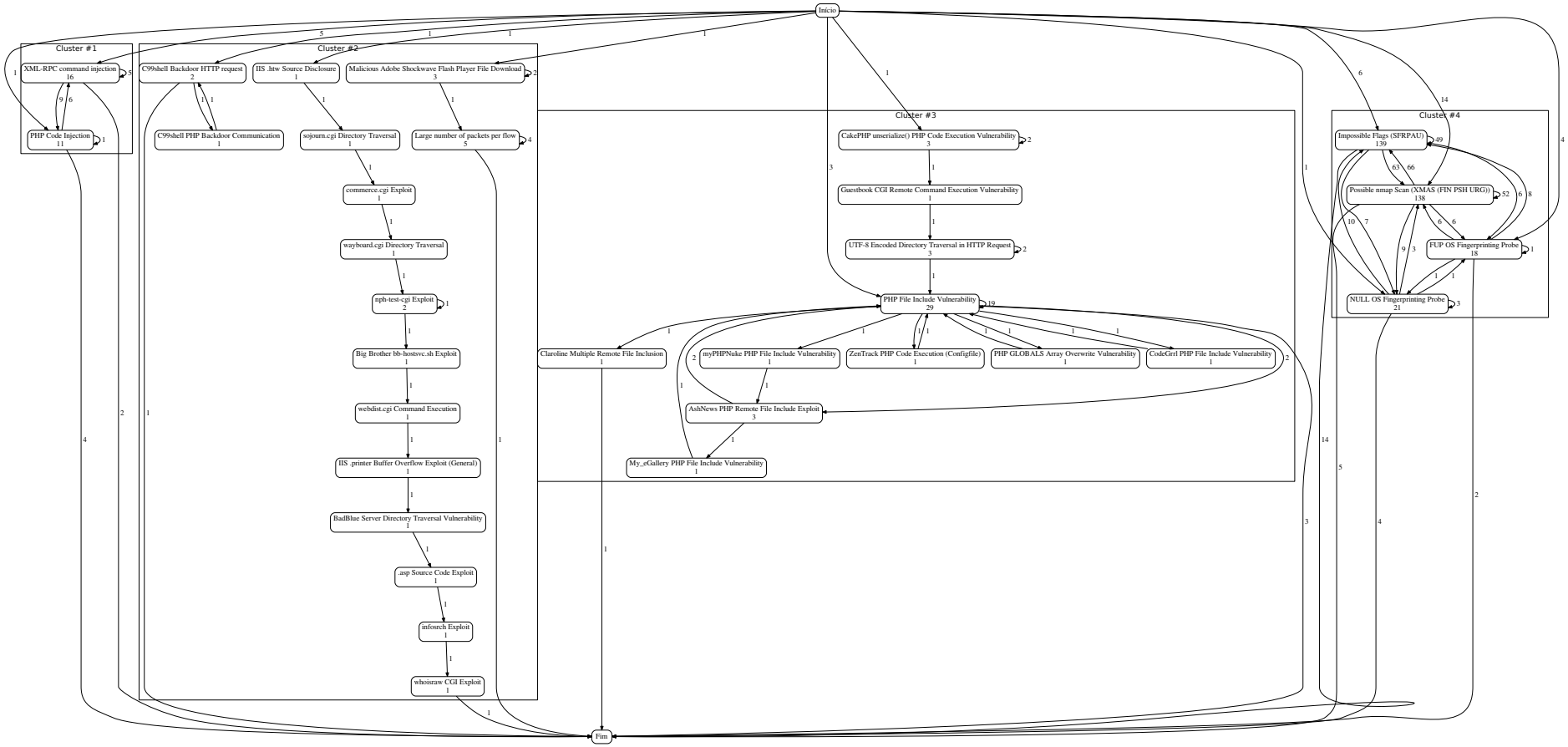


Figura 3.6 – Resultado da clusterização do modelo complexo da Figura 3.5.

4 EXPERIMENTOS E RESULTADOS

Este capítulo apresenta um estudo de caso que tem como objetivo avaliar a abordagem proposta neste trabalho. O estudo de caso foi organizado de acordo com alguns critérios que possivelmente podem influenciar os modelos de ataques gerados, como o número total de alertas, o número de atacantes distintos e o número de assinaturas distintas. Primeiramente, é apresentado o conjunto de dados utilizado no estudo de caso. A seguir, as etapas da abordagem são executadas no conjunto de dados utilizando os critérios previamente estabelecidos, em que uma discussão é realizada acerca dos resultados obtidos.

4.1 Conjunto de dados

Para avaliar a abordagem proposta, são utilizados alertas gerados por um IPS que utiliza o método de detecção baseado em assinaturas. O IPS faz a proteção do perímetro da rede da Universidade de *Maryland*, localizada nos Estados Unidos, cuja a rede é formada por cerca de quarenta mil computadores. O IPS está implantado na borda da rede da Universidade de *Maryland* com à Internet e monitora tanto tráfego de rede de entrada, isto é, proveniente da Internet direcionado à Universidade, quanto tráfego de saída, proveniente da Universidade direcionado à Internet. Os alertas disparados pelo IPS da Universidade são registrados no formato CSV (*Comma-Separated Values*) e possuem dezessete atributos. Alguns dos atributos dos alertas são apresentados na Tabela 4.1.

Os alertas considerados para realizar o estudo de caso ocorreram entre os anos de 2011 e 2013. No entanto, devido a alguns problemas com relação ao servidor de armazenamento dos logs, alguns alertas referentes a determinados meses como os de janeiro, fevereiro e março do ano de 2012 não foram registrados. Além disso, devido a ausência de atualizações e modificações na criação de assinaturas no IPS da Universidade, alguns alertas, principalmente do ano de 2011, não possuem uma assinatura correspondente. A Tabela 4.2 sumariza o conjunto de dados de alertas da Universidade de *Maryland* nesses anos.

Por mostrar-se mais estável com relação aos erros ocorridos nos registros dos alertas, além de compreender grande parte do número total dos alertas dentre os três anos considerados - aproximadamente 74,40% do total de quase 84 milhões de alertas - o conjunto de dados de alertas de *Maryland* do ano de 2012 foi selecionado para realização do estudo de caso. A Tabela 4.3 apresenta a distribuição mensal dos alertas que ocorreram no ano de 2012 na Universidade.

Considerando os alertas de *Maryland* que ocorreram durante o ano de 2012, di-

Tabela 4.1 – Exemplos de alguns atributos dos alertas de *Maryland*.

Atributo	**Descrição**
id	Identificador único de cada alerta.
timestamp	Registro de data e horário de quando o alerta foi gerado pelo IPS.
signature	A assinatura do alerta que identifica qual a violação executada pelo atacante.
category	A categoria de ataque em que o alerta se encontra. Alguns dos possíveis valores no campo são: *Attacks-Vulnerabilities*, *Attacks-Exploits*, *Reconnaissance*, *Virus*, *Spyware*, entre outros.
sport	Porta de origem utilizada pelo atacante para executar a violação.
dport	Porta de destino que identifica qual serviço foi alvo do possível ataque. Por exemplo, um ataque a um servidor HTTP terá como porta de destino o valor 80. Já um ataque a um servidor FTP terá como porta de destino o valor 21.
severity	Representa a severidade do alerta. Os níveis de severidade podem assumir os valores de 1 a 4. Quanto menor o valor, maior a severidade do ataque.
sip_hash	Identifica o endereço IP do *host* responsável por gerar o alerta.
dip_hash	Identifica o endereço IP do *host* alvo do possível ataque.
sip_asn	Identifica o ASN de origem do alerta. O valor igual a 1 no campo indica que o ataque teve origem na Universidade de *Maryland* e foi direcionado à Internet.
dip_asn	Identifica o ASN de destino do alerta. O valor igual a 1 no campo indica que o ataque teve origem na Internet e foi direcionado à Universidade de *Maryland*.

Tabela 4.2 – Resumo dos alertas de *Maryland* entre os anos de 2011 e 2013.

Conjunto de dados	**Mês inicial**	**Mês final**	**Total de alertas**	**Alertas sem assinatura**
Maryland 2011	Janeiro	Outubro	334.228	280.500
Maryland 2012	Abril	Dezembro	62.276.658	265.677
Maryland 2013	Janeiro	Março	20.075.872	12.309
Total	–	–	83.697.499	558.486

ferentes períodos para análise podem ser considerados. O período delimita o escopo do estudo de caso na investigação dos alertas e pode ser explorado para obter uma visão mais abrangente da rede em relação aos ataques. Por exemplo, o estudo de caso pode ser realizado considerando os alertas que ocorreram durante todo o ano ou apenas nos alertas

Tabela 4.3 – Resumo dos alertas de *Maryland* do ano de 2012 organizados por mês.

Mês	**Total de alertas**	**Alertas sem assinatura**
Abril	289.014	241.625
Maio	343.554	12.270
Junho	111.264	4.562
Julho	91.538	0
Agosto	88.438	0
Setembro	114.188	0
Outubro	4.402.686	0
Novembro	39.579.108	4.223
Dezembro	17.256.868	2.997

que ocorreram durante um mês ou dia, e assim por diante. Além disso, pode-se optar por realizar a análise apenas em determinados períodos específicos do ano, por exemplo, o período que corresponde ao semestre letivo ou o período que corresponde as férias. Devido a tantas possibilidades, é preciso definir qual será o escopo de análise dos alertas e quais serão os períodos do ano que serão analisados.

No estudo de caso realizado, o período de tempo para análise dos alertas foi definido como um dia. Esse período coincide com o intervalo de tempo t de ocorrência dos alertas que é utilizado como parâmetro para agrupar os alertas na abordagem. A escolha deve-se principalmente ao fato de que em um cenário real, considerando esse período, o administrador da rede pode ter conhecimento sobre os ataques que ocorrem diariamente contra a rede. Desse maneira, o período de tempo de 1 dia se mostra adequado para realização do estudo de caso.

Com o período de tempo para análise dos alertas definido, é preciso estabelecer quais os dias do ano serão selecionados para a realização do estudo de caso. Para evitar realizar a escolha de forma aleatória, três critérios para seleção dos dias foram considerados. Os critérios definidos têm como objetivo representar três situações distintas em que em altos números, podem implicar em conjuntos de alertas difíceis de serem interpretados pelo administrador da rede, além de poder influenciar nos modelos produzidos:

- **Por número de alertas**: Organizando os alertas por dia, será selecionado os dias do ano que apresentam as maiores quantidades de alertas disparados.
- **Por número de atacantes distintos**: Organizando os alertas por dia, será selecionado os dias do ano que apresentam os maiores números de atacantes distintos, ou seja, os maiores números de endereços IP de origem distintos nos alertas.
- **Por número de assinaturas distintas**: Organizando os alertas por dia, será selecionado os dias do ano que apresentam os maiores números de assinaturas distintas nos alertas.

O estudo de caso que aborda os critérios de seleção dos alertas nesses dias é apresentado nas próximas seções.

4.2 Estudo de caso

Nesta seção, o resultados obtidos no estudo de caso com base nos critérios definidos anteriormente são apresentados. Em cada um dos casos, as etapas da abordagem proposta são aplicadas e apresentadas juntamente com a análise dos resultados. Por apresentar um volume maior de alertas, apenas alertas oriundos de tráfego direcionados à Universidade foram considerados. Além disso, os parâmetros para avaliar a complexidade de um modelo G com V vértices e E arestas foram definidos como:

$$\text{É complexo}(G) = \begin{cases} \text{Não}, & \text{se } |V| \leq 12 \text{ ou } S > Y_k \\ \text{Sim}, & \text{se } |V| > 25 \text{ ou } S < Y_k \end{cases} \tag{4.1}$$

- Valor de limiar $Y_k = (0{,}0215 \times (\text{número de vértices do modelo})) + 0{,}0165$

Na determinação do valor de limiar de simplicidade, modelos com diferentes números de vértices e arestas foram gerados e classificados a partir de alertas disparados durante todo o ano de 2012, utilizando a agregação **um-para-muitos** com um intervalo $\boldsymbol{t = 1}$ **dia**. Nos modelos complexos, o algoritmo de clusterização hierárquica utilizado foi o método de ***Ward*** e a distância utilizada foi a dissimilaridade de **Jaccard**. Nas próximas seções, a análise e os resultados do estudo de caso seguindo os critérios definidos são apresentados.

4.2.1 Por número de alertas

Os IDSs são dispositivos extensivamente utilizados como uma das camadas de proteção de uma rede e de sistemas de computadores a fim de evitar e mitigar as consequências causadas por violações de segurança. No entanto, é comum um IDS disparar grandes quantidades de alertas que acabam sobrecarregando os administradores em suas atividades. Dessa maneira, o primeiro estudo de caso tem como objetivo avaliar a abordagem proposta quanto ao grande volume de alertas que são gerados pelos IDSs. Nesse sentido, foram selecionados no conjunto de dados os dois dias do ano que tiveram o maior número de alertas registrados. Os dias selecionados foram 06/11/2012 e 14/12/2012 e a quantidade de alertas disparados em cada um dos dias é apresentada pela Tabela 4.4.

Ambos os dias mostram efetivamente que é inviável realizar a análise manual de uma quantidade tão grande de alertas por um operador humano. Isso reitera a necessidade de abordagens automatizadas para auxiliar o administrador da rede na análise dos alertas.

Tabela 4.4 – Os dias do ano de 2012 com a maior quantidade de alertas disparados.

Dia	**Número total de alertas**
06/11/2012	26.873.302
14/12/2012	3.173.682

O primeiro passo da abordagem é agrupar os alertas utilizando a agregação um-para-muitos com base no IP de origem do alerta. O objetivo é formar grupos de alertas que estão associados às ações de um mesmo atacante (Seção 3.1.1). Em seguida, os grupos de alertas que possuem um único alerta e os grupos com alertas associados a uma única assinatura, considerados ruídos pela abordagem, são filtrados. As etapas de agregação e filtragem produzem os seguintes resultados em 06/11/2012:

- Número de eventos: 329.264
- Número de *cases*: 9.298
- Número de atividades: 66

De maneira similar, os resultados obtidos em 14/12/2012 são:

- Número de eventos: 904
- Número de *cases*: 108
- Número de atividades: 6

Nota-se que em ambos os casos, mais de 98% dos alertas foram filtrados. A redução mostra que a grande maioria dos ataques nesses dias executam apenas um tipo de violação e são finalizados, diferente de um ataque multiestágio em que os ataques evoluem por etapas e é o foco da abordagem. Existem algumas possíveis explicações para esse comportamento: i) os alertas são falsos positivos decorrentes, por exemplo, de um algum erro de configuração, ii) as outras etapas que compõem o possível ataque não foram detectadas pelo IPS ou iii) os atacantes utilizaram técnicas de IP *spoofing* impossibilitando identificar outros alertas do mesmo atacante. Contudo, apesar da grande redução, o dia 06/11/2012 ainda apresenta um número bastante grande de alertas, com cerca de 329 mil alertas.

A próxima etapa da abordagem é bastante direta. Os grupos formados são convertidos para o formato de log de eventos e, a seguir, o log é utilizado para a descoberta do modelo de ataque (Seção 3.1.2 e 3.1.3). O modelo de ataque descoberto por meio dos alertas disparados no dia 06/11/2012 é apresentado pela Figura 4.1. O modelo gerado apresenta as seguintes características:

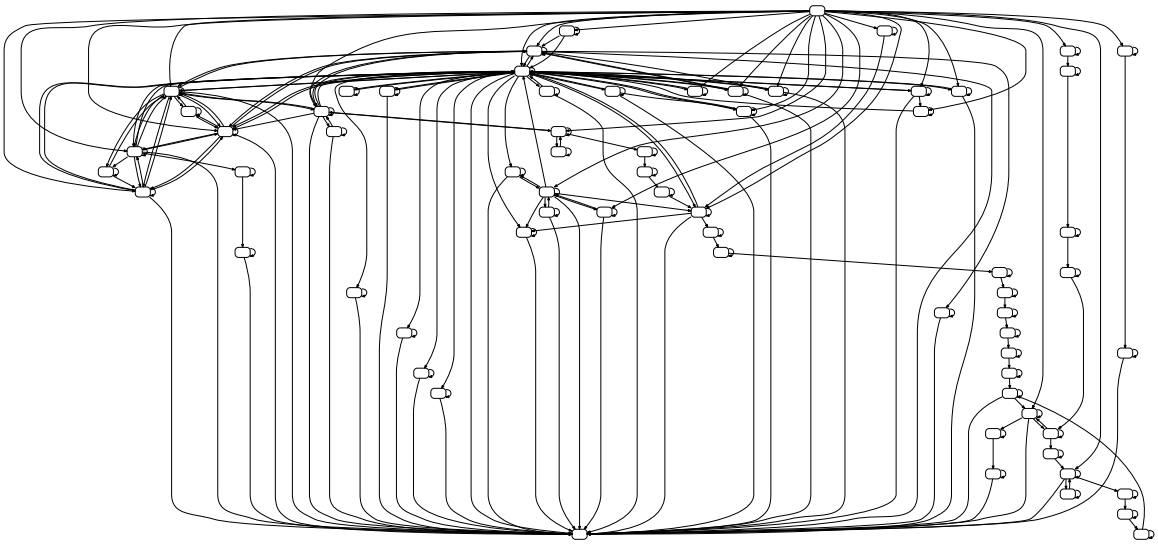


Figura 4.1 – Modelo de ataque dos alertas ocorridos no dia 06/11/2012.

- Número de vértices: 68
- Número de arestas: 251

Observando o modelo é possível verificar, no contexto desse estudo de caso, que ele é complexo, já que o número de vértices do modelo é maior do que o valor de limiar máximo tolerado que é de 25. Além disso, o modelo apresenta simplicidade igual a 0,27 que é menor que o valor de limiar permitido 0,55. Nota-se que não é possível realizar a análise do modelo devido a sua grande quantidade de arestas e vértices que produz um número muito grande de fluxos de ataque para investigação. Desse modo, o modelo gerado não pode ser utilizado para auxiliar o administrador a entender melhor as estratégias de ataques que estão sendo empregadas na rede. A solução para o problema é, portanto, clusterizar o modelo em modelos menores e mais simples utilizando as técnicas de clusterização hierárquica (Seção 3.1.4).

Aplicando o Algoritmo 3.2 no modelo complexo com os parâmetros da clusterização hierárquica definidos no início do seção, o modelo complexo é clusterizado em 25 modelos. A Figura 4.2 apresenta um exemplo de um dos modelos após a clusterização. O *cluster* apresenta as seguintes características:

- Número de eventos: 396
- Número de *cases*: 9
- Número de atividades: 6

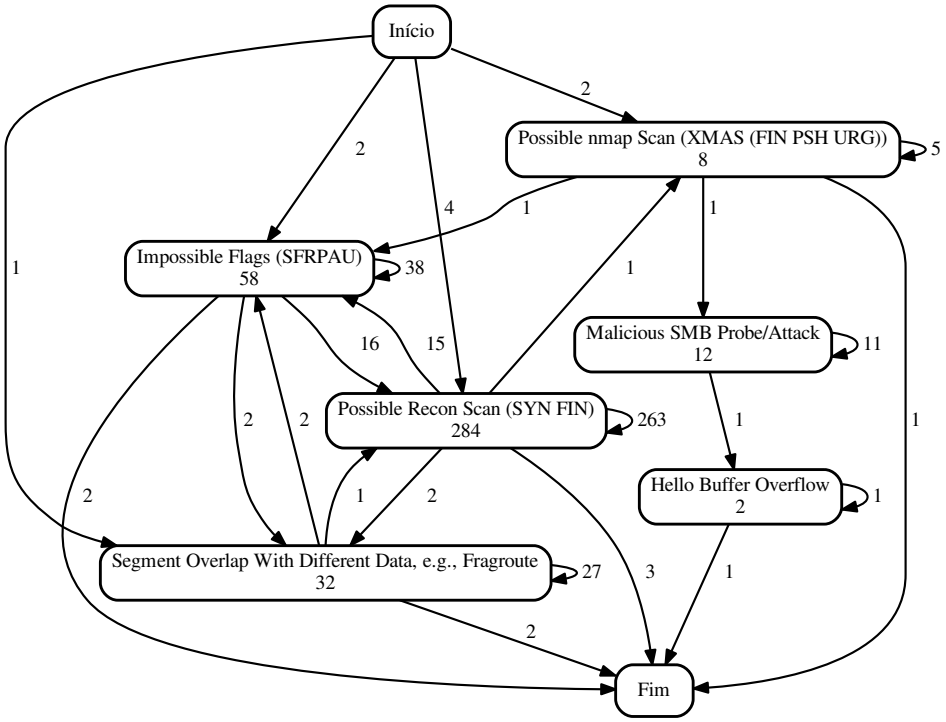


Figura 4.2 – Exemplo de um modelo (*cluster*) do dia 06/11/2012.

Uma vez que os modelos após a clusterização não são complexos, agora a análise das estratégias de ataques que ocorreram no dia 06/11/2012 é possível de ser realizada. Por exemplo, a análise do modelo da Figura 4.2 indica que:

- Nesse dia, os ataques foram iniciados com uma entre as quatro violações: i) *Segment Overlap With Different Data, e.g., Fragroute*; ii) *Impossible Flags (SFRPAU)*; iii) *Possible Recon Scan (SYN FIN)* e iv) *Possible nmap Scan (XMAS (FIN PSH URG))*. Cada uma das violações levam a um fluxo de ataque, isto é, um caminho diferente no modelo.

- Em i), o ataque inicia com um *Segment Overlap With Different Data, e.g., Fragroute*, uma técnica de evasão de IDS. Técnicas de evasão de IDS têm como objetivo modificar um ataque para que ele não seja detectado por um IDS. Uma das formas de realizá-la é por meio da sobreposição dos fragmentos (*Segment Overlap*) dos pacotes TCP [16]. A ideia é enviar o ataque em fragmentos, em que fragmentos enviados posteriormente irão sobrescrever partes dos fragmentos já recebidos com partes do ataque. Caso o pacote não seja reagrupado adequadamente no IDS, o ataque passará despercebido. Após executar a violação, o fluxo de ataque se divide em dois caminhos, um conduz a ii) e o outro conduz a iii).

- Em ii), iii) e iv), todas as três violações estão associadas a ataques de reconhecimento. Ataques de reconhecimento são técnicas bastante conhecidas usadas em

fases pré-ataque para coletar informações sobre a vítima e ser capaz de explorá-las. As violações em questão são técnicas de *port scan* utilizadas pelos atacantes com o objetivo de sondar a vítima por portas TCP abertas. Isso indica que um reconhecimento das possíveis vulnerabilidades das vítimas está sendo realizado antes da execução dos ataques.

- Em iv), após o *port scan*, é realizado um *Malicious SMB Probe/Attack* seguido por um *Hello Buffer Overflow*. O ataque *Malicious SMB Probe/Attack* tenta explorar uma vulnerabilidade no protocolo SMB (*Server Message Block*) utilizado para fornecer acesso compartilhado a arquivos. No Sistema Operacional (SO) *Windows*, caso a vulnerabilidade seja explorada com sucesso, a vulnerabilidade permite a execução remota de código, como relatado pelo boletim de segurança da Microsoft [66]. Por outro lado, o ataque *Hello Buffer Overflow* ocorre quando um programa aloca uma quantidade fixa de memória para armazenamento de dados mas recebe mais dados do que o esperado. Caso a vulnerabilidade exista, o atacante pode utilizar os dados adicionais para sobrescrever partes adjacentes da região de memória alocada, por exemplo, sobrescrever um ponteiro de retorno de uma pilha, para que o controle de programa salte para o código malicioso inserido pelo atacante (um vírus, *malware* ou *backdoor*) [16]. A princípio, esses dois ataques não possuem uma relação direta. No entanto, como mencionado acima, ambos podem ser usados para explorar vulnerabilidades e permitir a execução remota de códigos na vítima.

Por meio da análise das estratégias de ataques apresentadas pelo modelo clusterizado, é possível notar que os ataques realizados não estão sendo executados de forma aleatória, mas sim de forma planejada. Os fluxos envolvendo os ataques seguem a sequência de etapas que normalmente compõem os ataques: i) etapa de reconhecimento, ii) comprometimento da vítima e iii) manutenção do acesso a vítima (por um *backdoor* ou execução remota de código) [16]. Portanto, uma atenção maior por parte do administrador da rede deve ser tomada quanto aos ataques executados nesse dia.

Diferente do modelo gerado pelos alertas que ocorreram no dia 06/11/2012, o modelo descoberto pelos alertas do dia 14/12/2012 apresenta 8 vértices e 28 arestas. O modelo não é complexo, uma vez que no contexto do estudo de caso, possui um valor de limiar mínimo de vértices tolerado. Além disso, o modelo apresenta simplicidade é igual a $0{,}28 > 0{,}18 = Y_k$ (Equação 4.1). Logo, o modelo não precisa ser clusterizado para análise e pode ser apresentado diretamente para o administrador da rede. O modelo de ataque é apresentado pela Figura 4.3.

No modelo (Figura 4.3) nota-se que dois ataques ocorrem com uma frequência maior em relação aos outros ataques. São eles o *Zero Access Trojan Communication Attempt* (609 vezes) e *Version Request (udp)* (193 vezes), como mostra o gráfico com a

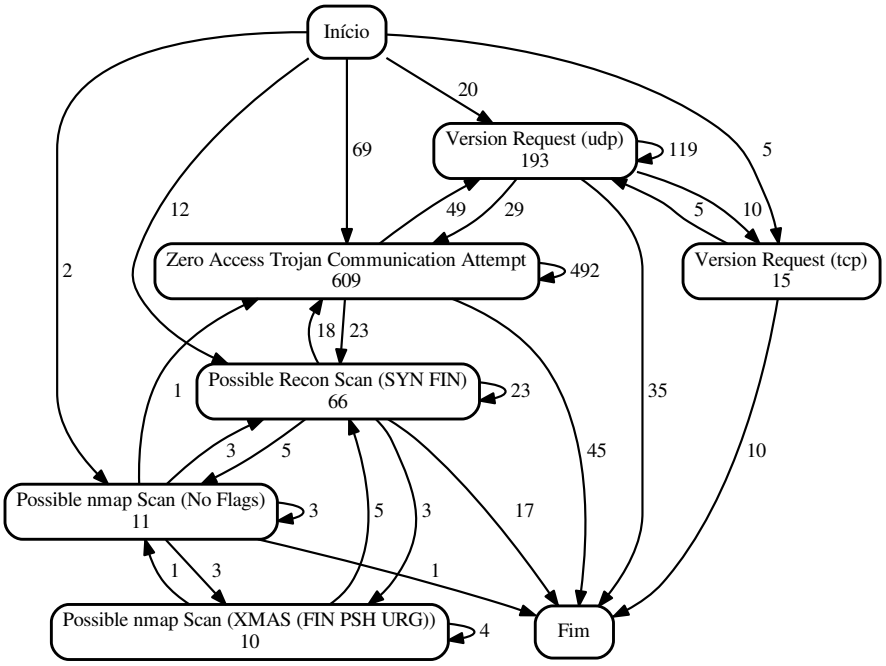


Figura 4.3 – Modelo de ataque dos alertas ocorridos no dia 14/12/2012.

frequência dos ataques da Figura 4.4. O primeiro diz respeito a um ataque do tipo Cavalo de Troia *ZeroAccess* que tem como objetivo criar um *backdoor* no sistema e simultaneamente se manter indetectado pela vítima. O ataque faz uso de um *rootkit*, um conjunto de programas maliciosos furtivos projetados para ocultar a existência de determinados programas e evidências das atividades do atacante em suas vítimas [16]. O segundo ataque é um ataque de reconhecimento que tem como objetivo obter a versão do servidor DNS (*Domain Name System*) a fim de determinar se o servidor é vulnerável a ataques do tipo *Buffer Overflow* ou DDoS (*Distributed Denial of Service*).

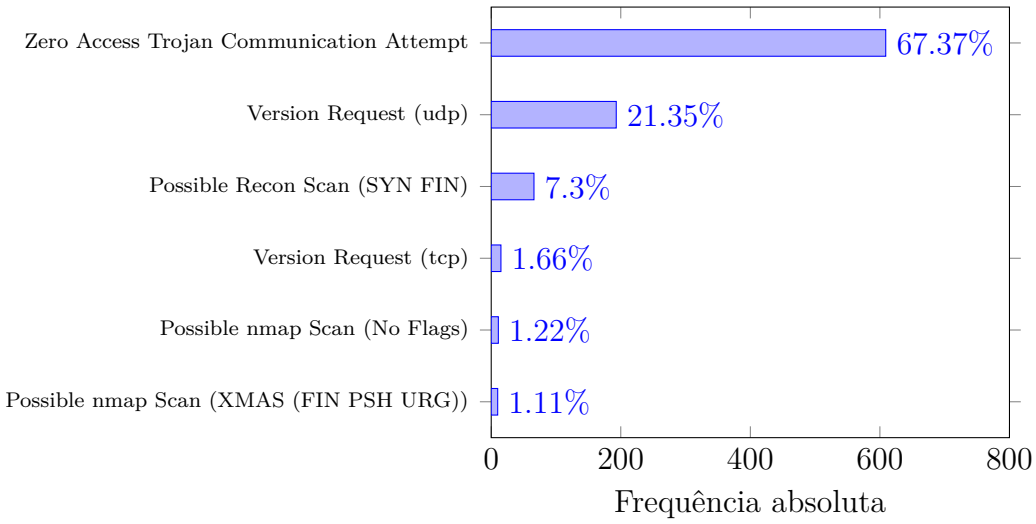


Figura 4.4 – Frequência das atividades do modelo da Figura 4.3.

A análise da frequência das atividades apresentadas pelo modelo mostra que am-

bos os ataques compreendem quase 89% dos ataques realizados no dia, em que o restante envolve alguns ataques de reconhecimento. Isso pode indicar que, nesse dia, o principal objetivo dos ataques era infectar as vítimas por meio do *ZeroAccess* e criar um *backdoor* que poderia ser utilizado para a execução de outros ataques futuramente. A análise do modelo apresenta um cenário que merece maior atenção do administrador da rede, possibilitando que medidas preventivas sobretudo contra *malwares* e proteção do servidor DNS sejam acionadas.

4.2.2 Por número de atacantes distintos

O segundo critério utilizado no estudo de caso representa uma situação em que há um grande número de atacantes tentando comprometer a rede. Nesse sentido, foram selecionados no conjunto de dados os dias do ano com o maior número de atacantes distintos. Os dias selecionados foram 06/11/2012, 14/12/2012 e 13/12/2012. O número de atacantes distintos em cada um dos dias é apresentado pela Tabela 4.5.

Tabela 4.5 – Os dias do ano de 2012 com o maior número de atacantes distintos.

Dia	**Número de atacantes distintos**	**Número total de alertas**
06/11/2012	1.586.897	26.873.302
14/12/2012	728.349	3.173.682
13/12/2012	562.267	2.060.202

Visto que os modelos dos dias 06/11/2012 e 14/12/2012 por representarem os dois dias do ano com os maiores números de alertas registrados foram analisados na seção anterior, o dia selecionado para o estudo de caso foi o dia 13/12/2012. O dia apresenta pouco mais de 562 mil atacantes.

A aplicação da agregação e da filtragem nos alertas do dia 13/12/2012 produziu os seguintes resultados:

- Número de eventos: 1.658
- Número de *cases*: 67
- Número de atividades: 8

De maneira similar ao que ocorreu com os dias com o maior número de alertas, após a primeira etapa da abordagem, grande parte dos alertas foram filtrados o que reduziu grande parte do número de atacantes. Isso significa que somente 67 grupos do total de grupos formados possuem mais de um tipo de assinatura. Na verdade, é possível observar que essa situação ocorre na maioria dos alertas da Universidade e, como mencionado anteriormente, pode ser causado por conta de técnicas de IP *spoofing* ou constituírem alertas falso positivos conforme indicam pesquisas como [32, 33].

A seguir, a próxima etapa da abordagem é executada produzindo o modelo não complexo apresentado pela Figura 4.5. O modelo gerado apresenta 10 vértices e 37 arestas e simplicidade igual a $0{,}27 > 0{,}23 = Y_k$. Analisando a frequência das atividades do modelo, nota-se que muitos ataques de reconhecimento foram executados no dia (*Possible Recon Scan (SYN FIN)* (790 vezes), *Possible nmap Scan (XMAS (FIN PSH URG))* (154 vezes), *Impossible Flags (SFRPAU)* (132 vezes) e *Possible nmap Scan (No Flags)* (123 vezes)). De fato, esse comportamento é esperado já que ataques de reconhecimentos, em particular, ataques relacionados à *network scan* e *port scan*, tendem a ter um quantidade maior de alertas devido a sua natureza repetitiva, uma vez que várias faixas de endereços IP e vários intervalos de portas são testados pelos atacantes. Além disso, vários tentativas de ataque do Cavalo de Troia *ZeroAccess* foram executadas (400 vezes).

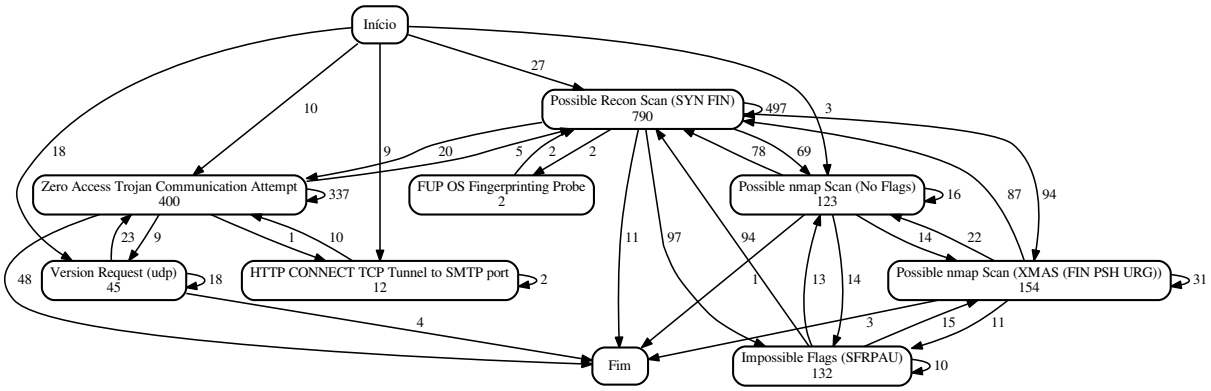


Figura 4.5 – Modelo de ataque dos alertas ocorridos no dia 13/12/2012.

O comportamento dos atacantes no dia 13/12/2012 é muito similar ao comportamento encontrado no dia 14/12/2012, o que indica que as mesmas estratégias de ataque podem ter sido executadas continuamente durante um período de tempo. O comportamento repetitivo durante os dias também pode indicar que os ataques executados estão sendo lançados por meio de ferramentas automatizadas configuradas para executar uma mesma sequência de ataque automaticamente durante os dias. Essas informações podem auxiliar o administrador da rede a avaliar melhor a situação para por em prática um plano de resposta contra as tentativas de ataque.

4.2.3 Por número de assinaturas distintas

Um dos principais objetivos e contribuições da abordagem são os modelos que apresentam as estratégias de ataque que estão sendo executadas contra a rede. Os modelos condensam as grandes quantidades de alertas gerados pelos IDSs e possibilitam a análise dos alertas de forma mais compreensível se comparado a análise manual. Dessa maneira, uma das principais preocupações na abordagem está em relação a complexidade dos modelos gerados.

Um dos fatores que influenciam a complexidade do modelo é o número de vértices que ele apresenta. Quanto maior for o número de vértices, maior torna-se o modelo e possivelmente mais complexo. Nesse sentido, foram selecionados no conjunto de dados os dois dias do ano que tiveram o maior número de assinaturas distintas. Os dias selecionados foram 23/09/2012 e 23/12/2012 como mostra a Tabela 4.6.

Tabela 4.6 – Os dias do ano de 2012 com o maior número de assinaturas distintas.

Dia	**Número de assinaturas**	**Número total de alertas**
23/09/2012	96	1.093
23/12/2012	94	13.789

Em ambos os dias, o número de assinaturas distintas é bastante elevado. Isso pode indicar que os ataques são bastantes diversos em relação às violações executadas e possivelmente compostos por várias etapas. Portanto, uma investigação minuciosa deve ser realizada. Aplicando a primeira etapa da abordagem, após o agrupar e filtrar os alertas que ocorreram em 23/09/2012 os seguintes resultados são obtidos:

- Número de eventos: 371
- Número de *cases*: 3
- Número de atividades: 87

Para o dia 23/12/2012 os resultados obtidos foram:

- Número de eventos: 2230
- Número de *cases*: 27
- Número de atividades: 81

Nota-se que após o agrupamento e filtragem dos alertas, em ambos os dias, a redução no número de assinaturas distintas foi pequena. No dia 23/09/2012, das 96 assinaturas distintas presentes nos alertas apenas 9 delas ocorreram exclusivamente nos grupos de alertas filtrados. Já no dia 23/12/2012, o mesmo ocorreu em apenas 13 assinaturas. Além disso, analisando os resultados obtidos, é possível notar um detalhe interessante em relação aos alertas do dia 23/09/2012. Nesse dia, somente três atacantes foram responsáveis pelos ataques e juntos foram executadas 87 violações distintas. Esse cenário pode indicar que: i) os ataques no dia são ataques multiestágio compostos por várias etapas e/ou ii) diferentes estratégias de ataque foram executadas no dia. Portanto, uma análise com mais detalhes deve ser realizada.

Nas próximas etapas da abordagem, realizando a descoberta do modelo nos alertas do dia 23/09/2012, tem-se como resultado o modelo de ataque complexo apresentado pela Figura 4.6. O modelo apresenta 89 vértices e 278 arestas e simplicidade igual a 0,32 < 1,93 = Y_k. Uma vez que os vértices no modelo representam as atividades, isto é, as assinaturas dos alertas que estão presentes em grande número no dia em questão, o modelo gerado é bastante complexo e sua análise é impraticável, logo, deve ser clusterizado. Aplicando a clusterização no modelo complexo, 11 *clusters* são formados. É interessante perceber que, no contexto da abordagem, a clusterização busca agrupar *cases* que sejam homogêneos em relação às suas atividades. No entanto, como no dia há apenas 3 *cases* para serem clusterizados, não há muitas opções para os agrupamentos. Dessa maneira, a clusterização é realizada com base no *timestamp* dos alertas de acordo com o Algoritmo 3.4. A Figura 4.7 apresenta o modelo resultante de um dos 11 *clusters* formados.

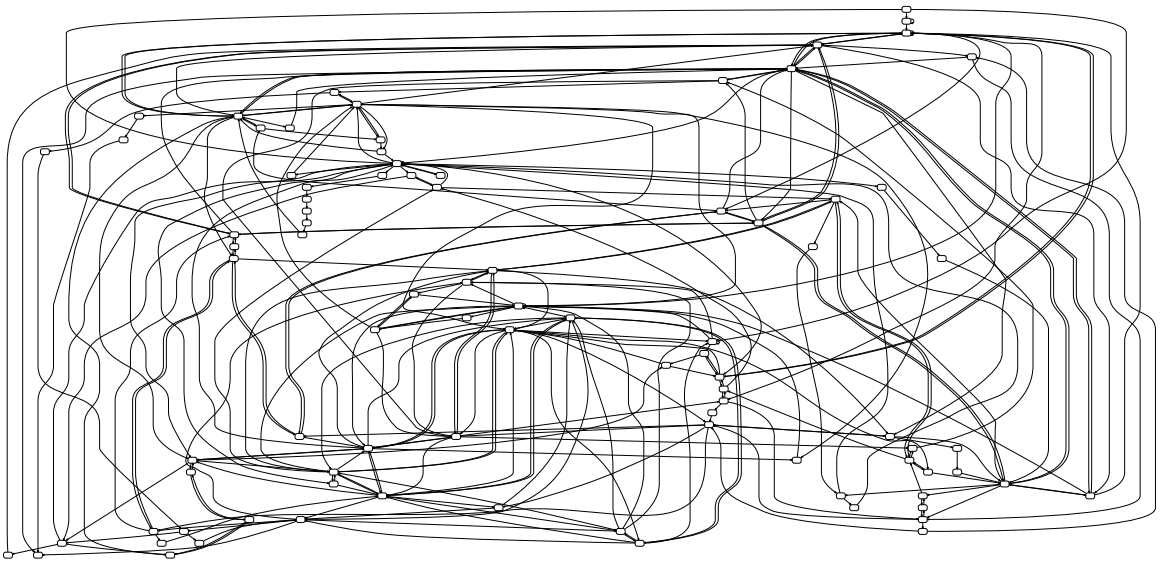


Figura 4.6 – Modelo de ataque dos alertas ocorridos no dia 23/09/2012.

Analisando o modelo, é possível perceber que diversas tentativas de *exploits* nos mais diversos serviços foram executadas. A princípio, não é possível perceber uma relação entre os ataques. Por exemplo, os ataques iniciam com um *YaBB.pl Exploit*, uma tentativa de explorar uma vulnerabilidade no programa de fórum YaBB (*Yet another Bulletin Board*). Em seguida, a tentativa de ataque é executada no programa de carrinhos de compras para comércio eletrônico *Commerce.CGI*. Logo depois, o mesmo ocorre no sistema de indexação e motor de busca *htsearch* e, por fim, o fluxo de ataque se divide. A partir desse fluxo, é possível perceber que os ataques que antes pareciam não ter relação entre si agora mostram que possuem algo em comum. Os ataques *htgrep Exploit*, *sojourn.cgi Directory Traversal*, *faxsurvey Exploit*, *Webstore Exploit*, *ph-test-cgi Exploit*, *pollit Exploit*, *carbo.dll Exploit*, *htsearch File Disclosure Exploit*, *webspirs Exploit*, *apexec.pl Exploit* e *pals-cgi Code Execution or File Read* estão relacionados a vulnerabilidades encontradas

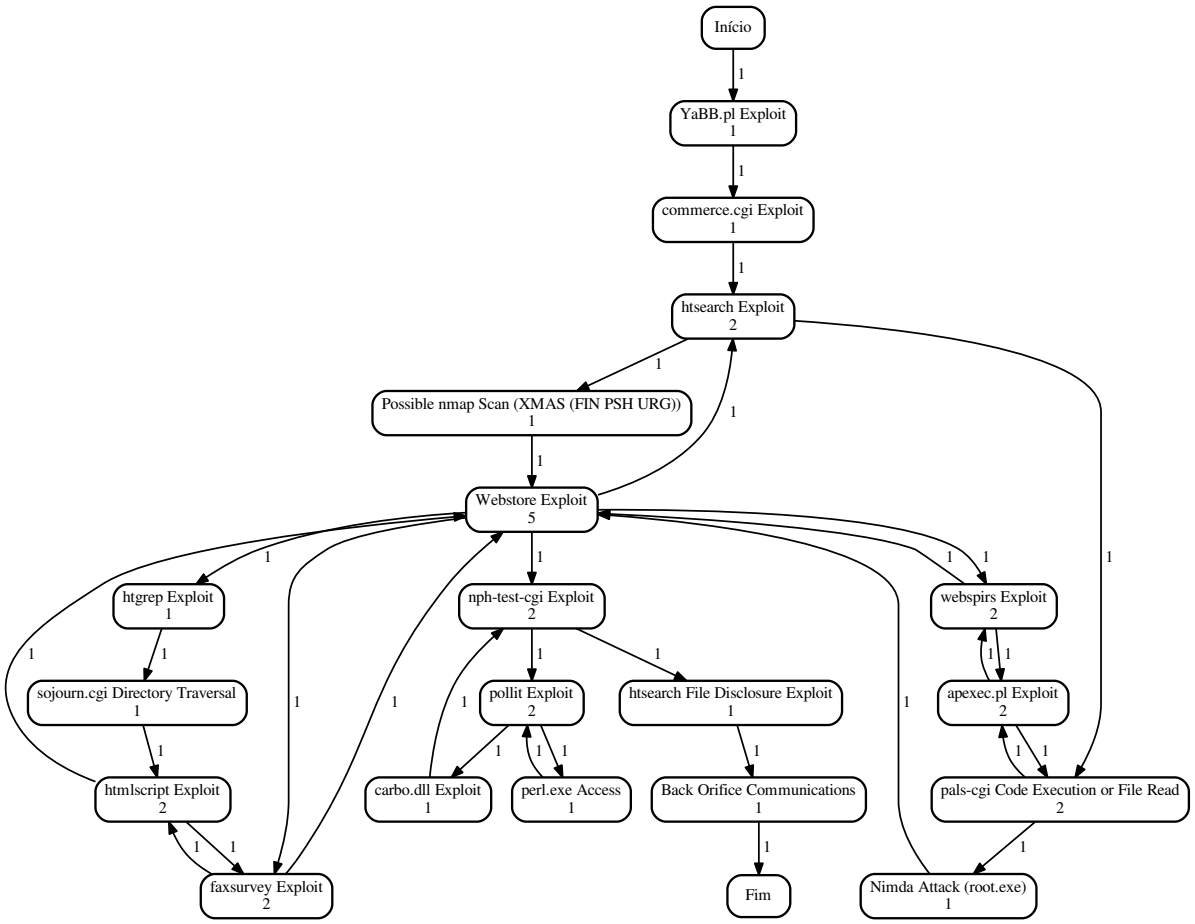


Figura 4.7 – Exemplo de um modelo (*cluster*) do dia 23/09/2012.

na tecnologia CGI (*Common Gateway Interface*). A tecnologia CGI é utilizada em servidores Web para permitir, por meio de *scripts*, a construção de páginas Web dinâmicas. A tecnologia não é uma linguagem de programação em si, mas sim uma interface que pode ser utilizada por outras linguagens, por exemplo, a *Perl*. Isso ajuda a explicar ataques como *Perl.exe Acces*, *YaBB.pl Exploit* e *apexec.pl Exploit* que utilizam a linguagem, e outros ataques a serviços que utilizam a tecnologia CGI como *carbo.dll Exploit*. A partir da análise do modelo, nota-se que o alvo dos ataques é um possível servidor Web que utiliza a tecnologia CGI para implementar algum de seus serviços.

Da maneira similar ao modelo do dia 23/09/2012, o modelo gerado pelos alertas do dia 23/12/2012 é bastante complexo como mostra a Figura 4.8. O modelo apresenta 87 vértices e 316 arestas e simplicidade igual a $0{,}27 < 1{,}88 = Y_k$. Uma vez que o modelo é complexo, é necessário realizar sua redução de complexidade por meio da clusterização. O resultado da clusterização divide o modelo complexo em 20 *clusters*. Nesse caso, é interessante perceber que dos 27 *cases* presentes no log, apenas 2 deles são responsáveis por deixar o modelo complexo como mostra o dendrograma da Figura 4.9.

No dendrograma, nota-se que o comportamento dos atacantes dos *cases* $(3, 5, 7, 8, 9,$

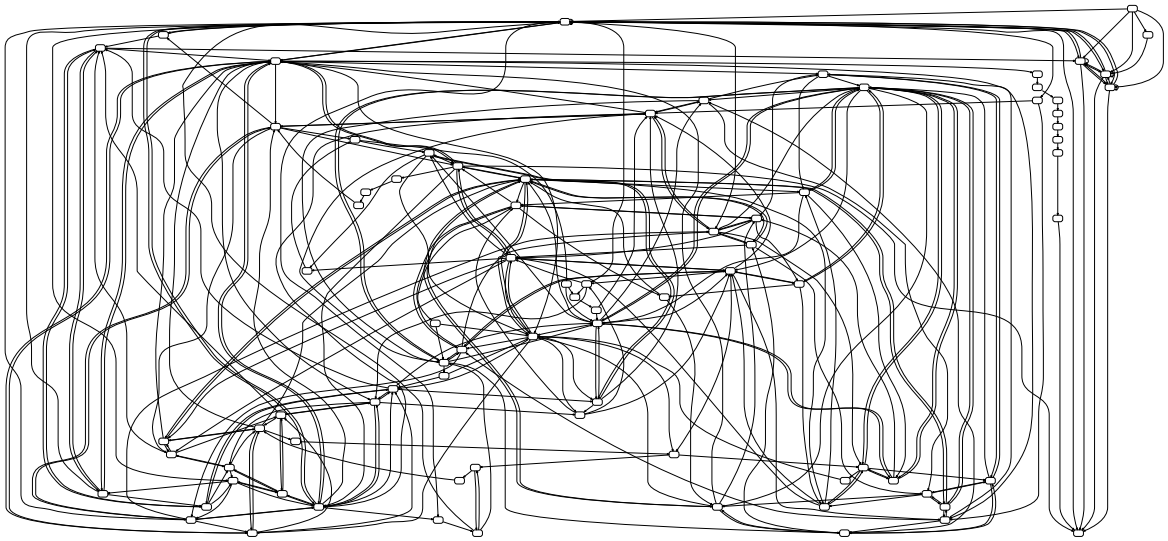


Figura 4.8 – Modelo de ataque dos alertas ocorridos no dia 23/12/2012.

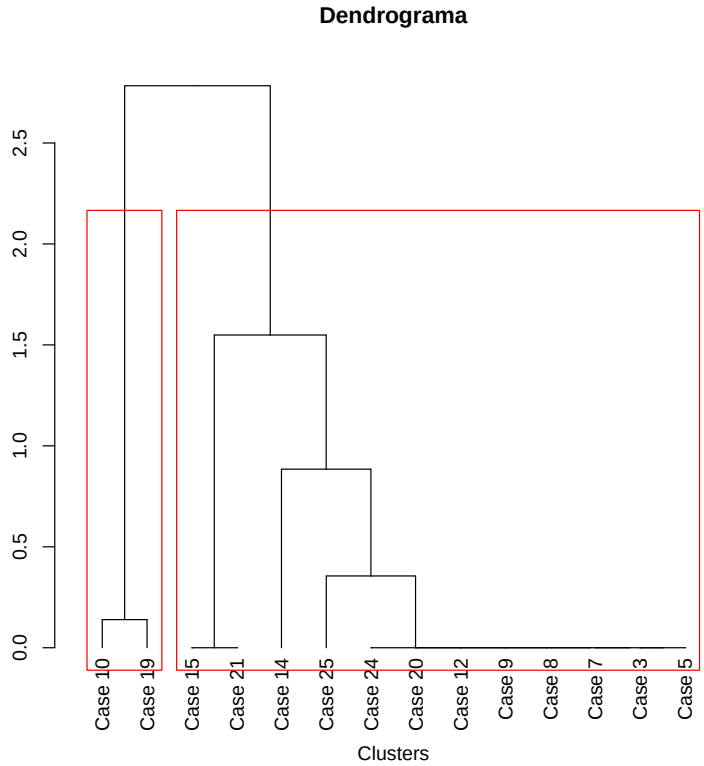


Figura 4.9 – Dendrograma da cluterização hierárquica do dia 23/12/2012.

12, 20 e 24) e (15 e 21) são idênticos, uma vez que os *cases* apresentam um nível de dissimilaridade 0 entre eles. Por outro lado, é possível ver que os *cases* 10 e 19 são muito dissimilares (nível acima de 2.5) em relação aos outros *cases* presentes no log. O motivo disso é que ambos os *cases* são responsáveis pela grande maioria dos ataques que ocorreram no dia e, portanto, o comportamento dos atacantes desses *cases* diferem-se dos demais. Logo, a clusterização hierárquica separa os *cases* do restante do grupo, e por

meio do Algoritmo 3.4, os *cases* são clusterizados com base no seu atributo de *timestamp*. A Figura 4.10 apresenta o modelo não complexo resultante de um dos 20 *clusters* formados, referente aos *cases* (3, 5, 7, 8, 9, 12, 14, 15, 20, 21, 24 e 25) em destaque (direita) do dendrograma (Figura 4.9).

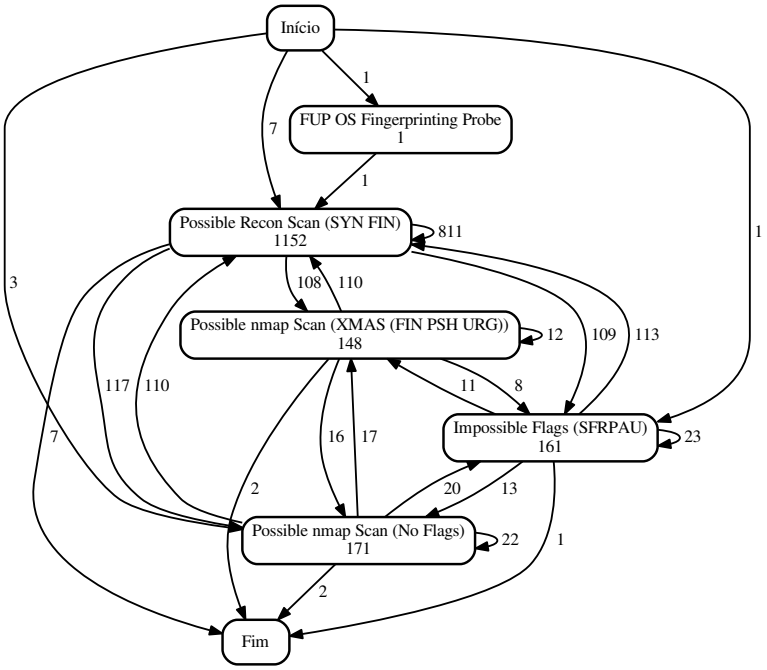


Figura 4.10 – Exemplo de um modelo (*cluster*) do dia 23/12/2012.

O *cluster* da Figura 4.10 apresenta os ataques de reconhecimento que ocorreram no dia. O modelo está de acordo com os outros dias do estudo de caso que ocorreram em dezembro em que os atacantes realizam ataques de reconhecimento a fim de comprometer efetivamente a vítima. No entanto, os *clusters* referentes aos alertas dos *cases* 10 e 19 não possuem tal comportamento. Em ambos os casos, um número muito grande de ataques distintos são executados, o que faz com seus modelos sejam complexos. Dessa forma, os *cases* são divididos com base no *timestamp* de seus eventos (Algoritmo 3.4) e cada modelo é dividido em outros 9 modelos. Juntos os dois *cases* (10 e 19) contribuem com 18 *clusters* dos 20 *clusters* produzidos pelo algoritmo.

Os modelos de ataque apresentados nesta seção mostram a importância de representações de alto nível como auxílio na análise dos alertas de IDS. Os modelos mostram a capacidade da abordagem proposta em sintetizar grandes quantidades de alertas em modelos com informações que dificilmente seriam obtidas realizando a análise manual. Por exemplo, o modelo de ataque da Figura 4.3 apresenta as estratégias de ataque que ocorreram em 14/12/2012. Nesse dia, mais de 3 milhões de alertas foram disparados e, devido a tantos alertas, a análise e investigação manual por um operador humano tornam-se impraticáveis. Além disso, os modelos gerados pela abordagem apresentam outros benefícios. Os modelos mostram os serviços da rede que têm sido alvo dos atacantes e merecem

uma maior atenção por parte do administrador. Por exemplo, em 14/12/2012, o modelo descoberto mostra que grande parte das tentativas de ataque realizadas no dia tinham como objetivo criar um *backdoor* nas vítimas por meio de um *malware*. Outro exemplo ocorre em 23/09/2012, em que pela análise do modelo da Figura 4.7, pode-se perceber que um servidor tem sido alvo dos atacantes que estão tentando explorar vulnerabilidades relacionadas à tecnologia CGI. Portanto, nota-se que as informações representadas pelos modelos podem ajudar o administrador da rede a acionar medidas preventivas contra as tentativas de ataque, como a instalação de antivírus e *patches* de segurança para a correção das vulnerabilidades, reforçando a segurança da rede. Por fim, quanto a complexidade dos modelos, a abordagem proposta mostrou-se capaz de, automaticamente, analisar os modelos quanto a sua complexidade, dividindo-os em modelos mais simples e bastante intuitivos para análise.

5 CONCLUSÃO

Com o crescente aumento no número de ameaças à segurança das redes de computadores e sistemas de informações, empresas e organizações têm investido em medidas protetivas com o objetivo de mitigar e evitar as consequências causadas por violações de segurança. IDSs são uma das tecnologias que têm sido utilizadas extensivamente com esse propósito. Os IDSs monitoram e detectam atividades intrusivas e geram alertas para um administrador da rede reportando a ocorrência de intrusões. Para adquirir uma visão geral da rede e obter informações a respeito dos ataques, os administradores da rede muitas vezes realizam a investigação manual dos alertas. Entretanto, IDSs podem gerar grandes quantidades de alertas, tornado a investigação manual uma tarefa onerosa e complexa.

Este trabalho abordou o problema da análise de grandes volumes de alertas de intrusão propondo uma solução composta por quatro etapas. Cada etapa da abordagem visa realizar uma transformação nos alertas de IDS que vão desde a manipulação dos alertas de baixo nível até sua representação em modelos visuais de alto nível. As etapas da abordagem utilizam técnicas da mineração de processos para minerar os alertas de intrusão com o objetivo de extrair informações relevantes sobre o comportamento dos atacantes. Então, a partir dos comportamentos observados, as estratégias de ataque multiestágio utilizadas pelos atacantes na tentativa de comprometer a rede são obtidas. As estratégias são apresentadas ao administrador da rede em um modelo de ataque, que é uma representação visual de alto nível que facilita a investigação e análise dos alertas.

Os modelos de ataque são uma das principais contribuições da abordagem proposta. Os modelos fornecem diversas informações que podem auxiliar o administrador a entender sobre os ataques que estão sendo empregados contra a rede e podem apoiar o administrador na tomada de ações e medidas protetivas contra eles. A análise dos modelos permite descobrir as sequências e dependências entre as etapas realizadas nos ataques, quais são os serviços da rede que são alvos dos ataques, além de fornecer métricas sobre a frequência dos ataques executados. As métricas são encorporadas no modelo e fornecem informações para o administrador da rede sobre os ataques que são mais recorrentes e os ataques que ocorrem com menos frequência. Além disso, por meio das frequências apresentadas nas arestas do modelo, é possível analisar quais são os fluxos de ataque mais executados. Assim, as métricas podem ser utilizadas pelo administrador da rede para auxiliar a priorizar as ações contra ataques iminentes.

Na abordagem, notou-se que a complexidade de alguns modelos gerados podem dificultar sua análise devido ao grande número de vértices e arestas que eles apresentam. Foi observado que em algumas situações, devido principalmente ao grande número de atividades distintas encontradas nos log de alertas, os modelos gerados podem se tornar

grandes e complexos, o que prejudica sua análise e compreensão sobre as estratégias de ataque executadas. Com o intuito de simplificar os modelos complexos, a abordagem propôs um algoritmo que tem como objetivo clusterizar os modelos complexos em modelos mais simples e compreensíveis. O algoritmo utiliza técnicas de clusterização hierárquica em conjunto com uma métrica que avalia a complexidade dos modelos e um valor de limiar. A métrica e o valor de limiar são utilizadas pelo algoritmo e guiam o processo de clusterização dos modelos. Ambos os valores são independentes do algoritmo proposto e outros métodos de quantificação podem ser utilizados se adaptando às preferências do administrador da rede. Foi definido uma métrica para avaliar a complexidade dos modelos e um método para definir o valor de limiar nos modelos utilizando uma regressão linear. Com base no dendrograma e nas métricas definidas, o algoritmo clusteriza os modelos complexos, discretizando os *cases* presentes no log de alertas e reorganizando-os em grupos homogêneos utilizando o método de *Ward* e a distância de Jaccard.

A abordagem foi avaliada por meio de um estudo de caso utilizando uma base de dados reais de alertas de IDS, provenientes da Universidade de *Maryland*. O estudo de caso foi desenvolvido de acordo com alguns critérios que têm influência nos modelos produzidos. Nesse sentido, foram selecionados alertas que ocorreram em períodos específicos do ano de 2012 em *Maryland* que atendessem um dos seguintes critérios: i) os dias do ano com a maior quantidade de alertas, ii) os dias do ano com o maior número de atacantes distintos e iii) os dias do ano com o maior número de assinaturas distintas. Entre os critérios utilizados no estudo de caso, alguns dias selecionados apresentaram uma grande quantidade de alertas. Apenas no dia 06/11/2012, por exemplo, foram gerados mais de 26 milhões de alertas. No estudo de caso, foi observado que em alguns dias selecionados, algumas estratégias de ataque observadas nos alertas eram formadas por um único tipo de ataque. Isso pode indicar que esses alertas são falsos positivos causados por algum erro de configuração ou que técnicas de IP *spoofing* foram utilizadas pelos atacantes, impossibilitando identificar grupos de alertas oriundos de um mesmo atacante. Além disso, alguns modelos dos dias selecionados se mostraram complexos e precisaram ser clusterizados. Em um dos casos, para reduzir a complexidade do modelo do dia 06/11/2012, o modelo teve que ser clusterizado em outros 25 modelos. Em outros casos, foi observado que poucos atacantes eram responsáveis por tornar os modelos complexos. Nesses cenários, a clusterização com base nos *cases* não foi possível de ser realizada e os modelos foram divididos com base no *timestamp* dos alertas (eventos) utilizando o algoritmo proposto.

De maneira geral, os resultados mostraram que os modelos resultantes são capazes de sintetizar grandes volumes de alertas e representar os comportamentos e as características das tentativas de ataque de forma intuitiva e compreensível. Portanto, o modelo pode ser utilizado por administradores de rede para auxiliar a análise dos alertas de IDS como uma alternativa para a investigação manual. Como sugestão para trabalhos futuros, pode-se investigar a influência de alertas falsos positivos no conjunto de alertas para

aprimorar o método de filtragem dos ruídos realizado na abordagem. Além disso, pode-se explorar outras formas de agregação e outros períodos do ano do conjunto de dados de *Maryland* para realizar uma comparação dos modelos produzidos, possibilitando que novas informações como padrões ou tendências nos ataques possam ser observados. Por fim, pode-se implementar outras formas de quantificação dos modelos complexos, utilizando abordagens de aprendizado de máquina como perceptrons de múltiplas camadas que podem ser treinados por meio de modelos previamente classificados e utilizados para classificar outros modelos de forma automatizada.

REFERÊNCIAS

[1] WANG, L. et al. An attack graph-based probabilistic security metric. In: ATLURI, V. (Ed.). *Data and Applications Security XXII*. Springer Berlin Heidelberg, 2008, (Lecture Notes in Computer Science, v. 5094). p. 283–296. ISBN 978-3-540-70566-6. Disponível em: <http://dx.doi.org/10.1007/978-3-540-70567-3_22>.

[2] EVERITT, B. S. et al. *Hierarchical Clustering*. John Wiley & Sons, Ltd, 2011. 71–110 p. ISBN 9780470977811. Disponível em: <http://dx.doi.org/10.1002/9780470977811.ch4>.

[3] AALST, W. van der. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. [S.l.]: Springer, 2011. ISBN 3642193447.

[4] NIST. *National Vulnerability Database*. 2015. [Acesso: Junho, 2015]. Disponível em: <https://web.nvd.nist.gov/view/vuln/statistics>.

[5] AL-MAMORY, S. O.; ZHANG, H. Intrusion detection alarms reduction using root cause analysis and clustering. *Computer Communications*, v. 32, n. 2, p. 419 – 430, 2009. ISSN 0140-3664.

[6] NING, P.; XU, D. Learning attack strategies from intrusion alerts. In: *Proceedings of the 10th ACM Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2003. (CCS '03), p. 200–209. ISBN 1-58113-738-9.

[7] AL-MAMORY, S. O.; ZHANG, H. A Survey on IDS Alerts Processing Techniques. *ISP'07 Proceedings of the 6th WSEAS international conference on Information security and privacy*, p. 69–78, 2007.

[8] LAGZIAN, S. et al. Frequent item set mining-based alert correlation for extracting multi-stage attack scenarios. In: *Telecommunications (IST), 2012 Sixth International Symposium on*. [S.l.: s.n.], 2012. p. 1010–1014.

[9] XUEWEI, F. et al. An Approach of Discovering Causal Knowledge for Alert Correlating Based on Data Mining. In: *Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on*. [S.l.: s.n.], 2014. p. 57–62.

[10] KORDY, B.; PIÈTRE-CAMBACÉDÈS, L.; SCHWEITZER, P. Dag-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, v. 13–14, n. 0, p. 1 – 38, 2014. ISSN 1574-0137.

[11] JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: A review. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 31, n. 3, p. 264–323, set 1999. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/331499.331504>.

[12] LESOT, M.-J.; RIFQI, M.; BENHADDA, H. Similarity measures for binary and numerical data: a survey. *Int. J. Knowl. Eng. Soft Data Paradigm.*, Inderscience Publishers, Inderscience Publishers, Geneva, SWITZERLAND, v. 1, n. 1, p. 63–84, dez 2009. ISSN 1755-3210. Disponível em: <http://dx.doi.org/10.1504/IJKESDP.2009.021985>.

[13] SHIREY, R. W. *Internet Security Glossary*. 2000. RFC 2828.

[14] ANDRESS, J. Chapter 1 - what is information security? In: ANDRESS, J. (Ed.). *The Basics of Information Security*. Boston: Syngress, 2011. p. 1 – 16. ISBN 978-1-59749-653-7. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9781597496537000013>.

[15] ISO/IEC 27000:2014(E). *Information technology - Security techniques - Information security management systems - Overview and vocabulary*. 2014.

[16] VACCA, J. *Computer and information security handbook*. Amsterdam: Morgan Kaufmann Publishers is an imprint of Elsevier, 2013. ISBN 9780123943972.

[17] PATEL, A.; QASSIM, Q.; WILLS, C. A survey of intrusion detection and prevention systems. *Information Management & Computer Security*, Emerald, v. 18, n. 4, p. 277–290, out 2010.

[18] STALLINGS, W.; BROWN, L. *Computer Security: Principles and Practice*. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007. ISBN 0136004245, 9780136004240.

[19] HUBBALLI, N.; SURYANARAYANAN, V. False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications*, v. 49, n. 0, p. 1 – 17, 2014. ISSN 0140-3664. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0140366414001480>.

[20] SCARFONE, K.; MELL, P. Intrusion detection and prevention systems. In: STAVROULAKIS, P.; STAMP, M. (Ed.). *Handbook of Information and Communication Security*. Springer Berlin Heidelberg, 2010. p. 177–205. ISBN 978-3-642-04116-7. Disponível em: <http://dx.doi.org/10.1007/978-3-642-04117-4_9>.

[21] GARCÍA-TEODORO, P. et al. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, v. 28, n. 1–2, p. 18 – 28, 2009. ISSN 0167-4048. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167404808000692>.

[22] VERWOERD, T.; HUNT, R. Intrusion detection techniques and approaches. *Computer Communications*, v. 25, n. 15, p. 1356 – 1365, 2002. ISSN 0140-3664. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0140366402000373>.

[23] LIAO, H.-J. et al. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, v. 36, n. 1, p. 16 – 24, 2013. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804512001944>.

[24] ROESCH, M. Snort - lightweight intrusion detection for networks. In: *Proceedings of the 13th USENIX Conference on System Administration*. Berkeley, CA, USA: USENIX Association, 1999. (LISA '99), p. 229–238. Disponível em: <http://dl.acm.org/citation.cfm?id=1039834.1039864>.

[25] SCHNEIER, B. *Attack Trees: Modeling security threats*. 1999. Dr. Dobb's Journal. Disponível em: <https://www.schneier.com/paper-attacktrees-ddj-ft.html>.

[26] SCHNEIER, B. *Secrets and lies: digital security in a networked world*. [S.l.]: John Wiley & Sons, 2011.

[27] PHILLIPS, C.; SWILER, L. P. A graph-based system for network-vulnerability analysis. In: *Proceedings of the 1998 Workshop on New Security Paradigms*. New York, NY, USA: ACM, 1998. (NSPW '98), p. 71–79. ISBN 1-58113-168-2.

[28] SWILER, L. et al. Computer-attack graph generation tool. In: *DARPA Information Survivability Conference amp; Exposition II, 2001. DISCEX '01. Proceedings*. [S.l.: s.n.], 2001. v. 2, p. 307–321 vol.2.

[29] WANG, L.; SINGHAL, A.; JAJODIA, S. Toward measuring network security using attack graphs. In: *Proceedings of the 2007 ACM Workshop on Quality of Protection*. New York, NY, USA: ACM, 2007. (QoP '07), p. 49–54. ISBN 978-1-59593-885-5. Disponível em: <http://doi.acm.org/10.1145/1314257.1314273>.

[30] JHA, S.; SHEYNER, O.; WING, J. Two formal analyses of attack graphs. In: *Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE*. [S.l.: s.n.], 2002. p. 49–63. ISSN 1063-6900.

[31] SINGHAL, A.; OU, X. Security risk analysis of enterprise networks using probabilistic attack graphs. In: . [S.l.]: National Institute of Standards and Technology (NIST) Interagency Report 7788, 2011.

[32] JULISCH, K.; DACIER, M. Mining intrusion detection alarms for actionable knowledge. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02*, ACM Press, New York, New York, USA, p. 366, 2002.

[33] JULISCH, K. Clustering intrusion detection alarms to support root cause analysis. *ACM Trans. Inf. Syst. Secur.*, ACM, New York, NY, USA, v. 6, n. 4, p. 443–471, nov 2003. ISSN 1094-9224.

[34] NING, P.; CUI, Y.; REEVES, D. S. Constructing Attack Scenarios through Correlation of Intrusion Alerts. *Proceedings of the 9th ACM conference on Computer and communications security*, p. 10, 2002.

[35] TREINEN, J.; THURIMELLA, R. A Framework for the Application of Association Rule Mining in Large Intrusion Detection Infrastructures. *Recent Advances in Intrusion Detection*, p. 1–18, 2006.

[36] LEE, S. et al. Real-time analysis of intrusion detection alerts via correlation. *Computers & Security*, v. 25, n. 3, p. 169–183, 2006. ISSN 0167-4048.

[37] ZURUTUZA, U. et al. Combined Data Mining Approach for Intrusion Detection. In: *SECRYPT*. [S.l.: s.n.], 2007. p. 67–73.

[38] SOLEIMANI, M.; GHORBANI, A. A. Critical Episode Mining in Intrusion Detection Alerts. *6th Annual Communication Networks and Services Research Conference (cnsr 2008)*, IEEE, p. 157–164, maio 2008.

[39] SADODDIN, R.; GHORBANI, A. A. An incremental frequent structure mining framework for real-time alert correlation. *Computers & Security*, Elsevier Ltd, v. 28, n. 3-4, p. 153–173, maio 2009. ISSN 01674048.

[40] DASIREDDY, S. et al. Alerts visualization and clustering in network-based intrusion detection. *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research - CSIIRW '10*, p. 1, 2010.

[41] YANG, L. et al. Alerts analysis and visualization in network-based intrusion detection systems. *Proceedings - SocialCom 2010: 2nd IEEE International Conference on Social Computing, PASSAT 2010: 2nd IEEE International Conference on Privacy, Security, Risk and Trust*, p. 785–790, 2010.

[42] LIU, L.; ZHENG, K.; YANG, Y. X. An intrusion alert correlation approach based on finite automata. *Proceedings - 2010 International Conference on Communications and Intelligence Information Security, ICCIIS 2010*, p. 80–83, 2010.

[43] NJOGU, H. W.; LUO, L. Using Alert Cluster to reduce IDS alerts. *Proceedings - 2010 3rd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2010*, v. 5, p. 467–471, 2010.

[44] TAHA, A. E. et al. Agent based correlation model for intrusion detection alerts. *2010 IEEE International Conference on Intelligence and Security Informatics*, p. 89–94, 2010.

[45] CIPRIANO, C. et al. Nexat: A history-based approach to predict attacker actions. In: *Proceedings of the 27th Annual Computer Security Applications Conference*. New York, NY, USA: ACM, 2011. (ACSAC '11), p. 383–392. ISBN 978-1-4503-0672-0.

[46] SAAD, S.; TRAORE, I. A semantic analysis approach to manage IDS alerts flooding. *Proceedings of the 2011 7th International Conference on Information Assurance and Security, IAS 2011*, p. 156–161, 2011.

[47] AHMADINEJAD, S. H.; JALILI, S.; ABADI, M. A hybrid model for correlating alerts of known and unknown attack scenarios and updating attack graphs. *Computer Networks*, Elsevier B.V., v. 55, n. 9, p. 2221–2240, jun 2011. ISSN 13891286.

[48] FATMA, H.; MOHAMED, L. A two-stage technique to improve intrusion detection systems based on data mining algorithms. In: *Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on*. [S.l.: s.n.], 2013. p. 1–6.

[49] ZOMLOT, L. et al. Aiding intrusion analysis using machine learning. In: *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*. [S.l.: s.n.], 2013. v. 2, p. 40–47.

[50] SPATHOULAS, G. P.; KATSIKAS, S. K. Enhancing IDS performance through comprehensive alert post-processing. *Computers and Security*, Elsevier Ltd, v. 37, p. 176–196, 2013. ISSN 01674048.

[51] ZONG, B. et al. Towards scalable critical alert mining. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2014. (KDD '14), p. 1057–1066. ISBN 978-1-4503-2956-9.

[52] CHEN, S.; LEUNG, H.; DONDO, M. Characterization of computer network events through simultaneous feature selection and clustering of intrusion alerts. *SPIE Sensing Technology + Applications*, v. 9121, p. 912107, 2014. ISSN 1996756X.

[53] GHASEMIGOL, M.; GHAEMI-BAFGHI, A. E-correlator: an entropy-based alert correlation system. *Security and Communication Networks*, v. 8, n. 5, p. 822–836, 2014. ISSN 1939-0122.

[54] THOMAS, C.; SHARMA, V.; BALAKRISHNAN, N. Usefulness of darpa dataset for intrusion detection system evaluation. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *SPIE Defense and Security Symposium*. [S.l.], 2008. p. 69730G–69730G.

[55] BERKHIN, P. A survey of clustering data mining techniques. In: KOGAN, J.; NICHOLAS, C.; TEBOULLE, M. (Ed.). *Grouping Multidimensional Data*. Springer Berlin Heidelberg, 2006. p. 25–71. ISBN 978-3-540-28348-5. Disponível em: <http://dx.doi.org/10.1007/3-540-28349-8_2>.

[56] GAN, G.; MA, C.; WU, J. *Data clustering: theory, algorithms, and applications*. [S.l.]: Siam, 2007. v. 20.

[57] XU, R.; WUNSCH D., I. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, v. 16, n. 3, p. 645–678, maio 2005. ISSN 1045-9227.

[58] JAIN, A. K.; DUBES, R. C. *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. ISBN 0-13-022278-X.

[59] SHALIZI, C. Distances between clustering, hierarchical clustering. *Lectures notes*, 2009.

[60] JACOBS, J.; RUDIS, B. *Data-driven Security: Analysis, Visualization and Dashboards*. [S.l.]: John Wiley & Sons, 2014.

[61] MIANI, R. S. et al. A practical experience on evaluating intrusion prevention system event data as indicators of security issues. In: *Reliable Distributed Systems (SRDS), 2015 IEEE 34th Symposium on*. [S.l.: s.n.], 2015. p. 296–305.

[62] HÄTÄLÄ, A. et al. Event data exchange and intrusion alert correlation in heterogeneous networks. In: *Proceedings of the 8th Colloquium for Information Systems Security Education (CISSE), Westpoint, NY, CISSE (June 2004 2004)*. [S.l.: s.n.], 2004. p. 84–92.

[63] CERT. *CA-2001-26 Nimda Worm*. 2001. [Acesso: Junho, 2015]. Disponível em: <https://www.cert.org/historical/advisories/CA-2001-26.cfm>.

[64] CVE. *Vulnerability Summary for CVE-2003-0352*. 2003. [Acesso: Abril, 2016]. Disponível em: <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2003-0352>.

[65] CVE. *Vulnerability Summary for CVE-2003-0818*. 2004. [Acesso: Abril, 2016]. Disponível em: <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2003-0818>.

[66] MICROSOFT. *Microsoft Security Bulletin MS11-019: Vulnerabilities in SMB Client Could Allow Remote Code Execution (2511455)*. 2011. [Acesso: Junho, 2015]. Disponível em: <https://technet.microsoft.com/library/security/ms11-019>.

TRABALHOS PUBLICADOS PELO AUTOR

Trabalhos publicados pelo autor durante o programa.

1. ALVARENGA, S. C.; ZARPELAO, B. B.; SOARES, V. N. G. J.. **A notification architecture for smart cities based on push technologies**. In: 2014 XL Latin American Computing Conference (CLEI), 2014, Montevideo. 2014 XL Latin American Computing Conference (CLEI), 2014. p. 1. (Qualis CC 2012, B4).

2. ALVARENGA, S. C.; ZARPELAO, B. B.; JUNIOR, S. B.; MIANI, R. S.; CUKIER, M.; **Discovering Attack Strategies Using Process Mining**. In: The Eleventh Advanced International Conference on Telecommunications (AICT), 2015, Brussels. (Qualis CC 2012, B1).